Thèse

intitulée

Résolution exacte et approchée de problèmes de multiflot entier et de multicoupe : algorithmes et complexité

présentée au

Conservatoire National des Arts et Métiers

pour obtenir le titre de

DOCTEUR

en spécialité

INFORMATIQUE

par

Cédric BENTZ

et soutenue publiquement le

20 novembre 2006

devant le jury composé de :

Marie-Christine COSTA Professeur au CNAM Paris	Directrice de thèse
Frédéric ROUPIN Maître de conférences au CNAM Paris	Co-encadrant
Dominique DE WERRA Professeur à l'EPFL, Lausanne, Suisse	Rapporteur
András SEBÖ Directeur de recherche CNRS au laboratoire Leibniz, Grenoble	Rapporteur
Cristina BAZGAN Professeur à l'Université Paris Dauphine	Examinateur
Pascal BERTHOMÉ Maître de conférences à l'IUT d'Orsay	Examinateur
Christophe PICOULEAU Professeur au CNAM Paris	Examinateur

À ma mère,
À mes grands-parents.

Remerciements

Je tiens ici à remercier un certain nombre de personnes qui ont, directement ou non, contribué à ce que cette thèse existe.

Je voudrais en premier lieu remercier Marie-Christine Costa, ma directrice de thèse, ainsi que Frédéric Roupin, qui a co-encadré cette thèse, pour m'avoir permis de découvrir le monde fascinant de la recherche, des graphes et des (multi-)flots et (multi-)coupes. J'ai pris la décision de me lancer dans cette grande aventure à la suite du stage de DEA que j'ai effectué au sein du laboratoire CEDRIC sous leur direction ; sans cette expérience positive, je n'aurais probablement jamais continué dans cette voie. Je les remercie également d'avoir su me mettre le pied à l'étrier, tout en me laissant l'autonomie qui m'était nécessaire. Je remercie enfin Marie-Christine pour son soutien amical lors de différents moments ou événements pénibles survenus pendant ces trois années de thèse.

J'adresse ensuite ma plus sincère reconnaissance au Professeur Dominique de Werra, qui a bien voulu être rapporteur de cette thèse, et qui m'a, au cours de nos différentes rencontres, enseigné l'importance du jarret (grillé ou braisé) dans tout travail de recherche digne de ce nom.

Monsieur le Professeur András Sebö m'a fait l'honneur d'accepter d'être rapporteur de ce travail, et je tiens à lui exprimer ma plus profonde gratitude pour l'attention qu'il y a portée.

Je tiens enfin à exprimer ma vive reconnaissance au Professeur Cristina Bazgan et à Pascal Berthomé, qui m'ont fait le plaisir de participer à mon jury. J'en profite aussi pour adresser un immense merci au Professeur Christophe Picouleau, avec qui j'ai passé de longues heures pendant ces trois années, que ce soit devant un tableau noir ou au comptoir d'un café, et dont l'amitié et les conseils m'ont été très précieux. Il a, en outre, accepté d'être membre de mon jury, et je l'en remercie.

S'il est une personne pour qui ces trois ans n'ont pas été faciles, je pense que c'est bien Sylvie, ma compagne, qui a eu à me supporter pendant tout ce temps. Je tiens à la remercier de tout mon cœur pour sa patience, son amour et sa compréhension. Elle a même accepté de relire certaines parties de ma thèse, et je lui en suis très reconnaissant.

Je souhaite également remercier ma famille pour son soutien. En particulier, je remercie vivement mon père, Jean-Paul, pour la patience avec laquelle il m'a écouté exposer, pendant des heures, les problèmes d'ordre "graphien" qui me tourmentaient. Grâce à l'attention qu'il a portée à mes bavardages, il a même réussi à acquérir une certaine compréhension de ce domaine, qui ne cesse de m'étonner. Il a aussi participé à la relecture de ma thèse et, avec l'aide précieuse de ma belle-mère Claudine, a participé à la relecture des différents articles rédigés en anglais dans le cadre de cette thèse ; je leur en suis reconnaissant à tous les deux. Je remercie ma sœur Katia pour les longues séances de discussion métaphysico-philosophique que nous avons partagées, et durant lesquelles j'en oubliais le reste. Je remercie enfin Julien et le reste de ma famille pour leurs encouragements, et en particulier ma tante Chantal, mon oncle Christian, mon cousin Pascal et ma cousine Sylvie.

Je tiens aussi à remercier François et Xavier, mes vieux compagnons de route, qui ont finalement renoncé à me concurrencer en ne tentant pas l'aventure doctorale avec moi (!), mais qui n'ont jamais douté que je m'en sortirais (à tort ou à raison !). Je ne peux pas non plus manquer de saluer mes compagnons IIEns, avec qui j'ai combattu Cthulhu et tant d'autres, au cours de séances de jeu épiques et délassantes... Merci donc à Camille, Elric (pardon, Arnaud !), Julien et Luc.

Quatre personnes ont partagé mon bureau du CNAM pendant ces trois ans, et je voudrais remercier chacune d'elles. Merci donc à Tatiana, qui, par sa gentillesse et son expérience, a contribué à rendre la chose moins effrayante et n'a cessé de me prodiguer ses conseils tout au long de ma thèse. Merci également à Marie, avec qui j'ai partagé mes angoisses et mes espoirs de jeune chercheur depuis trois ans, et dont la présence en tant que "jumelle" de thèse a été très importante pour moi. Merci à Aurélie, qui a apporté un peu de fraîcheur à notre bureau, et a toujours été partante pour partager nos pauses café et nos repas de bureau dans la bonne humeur. Enfin, merci à Nicolas, animateur en chef de notre bureau. J'ai eu l'occasion de l'encadrer en stage de Master (sur des problèmes de coupes dans les arbres, comme l'illustre la figure donnée en page suivante), et travailler avec lui a été un véritable plaisir. J'espère qu'il prendra pleinement conscience de ses capacités, et que nous aurons de nouveau l'occasion de travailler ensemble dans le futur.

J'en profite également pour adresser mes remerciements aux membres de mon groupe, Alain, Florence, Franck et Nicolas (Nico), qui ont contribué à mon équilibre mental par le biais de nos répétitions régulières (!), et qui ont eu la gentillesse de se proposer pour relire ma thèse. Je tiens à saluer en particulier la contribution de Nico, qui, avec courage, en a relu une partie. Je remercie d'ailleurs l'ensemble des relecteurs de cette thèse, qui ont fait un travail épatant. Si des fautes persistent, j'en porte l'entière responsabilité, ayant été celui qui centralisait leurs remarques.

Je ne peux pas manquer non plus de remercier l'ensemble de l'équipe *Op-*

timisation Combinatoire du laboratoire CEDRIC, qui m'a fait un excellent accueil, et au sein de laquelle il a été fort agréable de travailler : merci, donc, à Agnès, Alain B. et Alain F., Eric, Fethi, Karima et Sourour. Merci également à Dominique, qui a partagé avec enthousiasme nos sorties "bureau" à différentes reprises, et à mes camarades de DEA, Aurélie, Cathy, Guillaume, Julien, Laure, Nathalie et Fred, Olivier et Yasmin, avec qui j'ai eu l'occasion de passer quelques soirées. Je remercie aussi l'ensemble des membres du CEDRIC que j'ai côtoyés pendant ma thèse, et avec qui le contact a été agréable. Enfin, je remercie le Professeur Bernard Lemaire et Marianne Martella de la chaire de Recherche Opérationnelle du CNAM, au sein de laquelle mes trois années de monitorat ont été instructives et fort sympathiques.

Il me reste quelques remerciements un peu particuliers à adresser. Tout d'abord, je voudrais rendre hommage à Messieurs Patrick Teller et Thierry Guitard, mes professeurs de mathématiques en classes de Mathématiques Supérieures et de Mathématiques Spéciales respectivement, qui ont su, avec beaucoup de persévérance, m'inculquer le goût des preuves, de la chose formelle, et, plus que tout, de la rigueur. Pour tout cela, je les remercie.

Je souhaite également remercier l'ensemble des chercheurs qui ont su aiguiser ma curiosité et mon goût pour les graphes et l'optimisation combinatoire, et dont les travaux ont été une source permanente d'inspiration pour moi. Il serait trop long de les citer tous, mais il suffit de parcourir ce document pour en connaître les principaux.

Enfin, je voudrais exprimer ma reconnaissance toute particulière envers un groupe de cinq personnes dont la musique a contribué, pendant ces trois (et même ces dix) dernières années, à me porter vers l'avant, dans les moments agréables comme dans les autres. Merci à Thom, Colin, Jonny, Ed et Phil ; puisse leur musique continuer de m'accompagner.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Dans cette thèse, on s'intéresse aux aspects algorithmiques de deux problèmes combinatoires fondamentaux : le problème du multiflot entier maximum et le problème de la multicoupe minimum. Ces deux problèmes (et un certain nombre de leurs variantes) sont d'une grande importance à la fois théorique et pratique : d'une part, car leur résolution par des algorithmes efficaces reste encore assez mal comprise, d'autre part parce que ce sont des problèmes aux applications industrielles nombreuses, notamment dans la conception et l'exploitation de réseaux de télécommunications (routage, fiabilité, etc.).

Les premiers problèmes de flot et de coupe étudiés par la communauté scientifique, du point de vue algorithmique, sont les fameux problèmes du *flot maximum* et de la *coupe minimum*, dont les propriétés essentielles ont été découvertes, en particulier, par Ford et Fulkerson dans leur célèbre article fondateur de 1956 [65]. De nombreuses recherches ont, par la suite, permis d'améliorer sensiblement à la fois la compréhension de ces problèmes et les méthodes de résolution [5]. Les deux problèmes que l'on se propose d'étudier dans cette thèse constituent des *généralisations* de ces deux problèmes classiques ; néanmoins, de nombreuses propriétés avantageuses qui leur sont associées ne se conservent pas. Ainsi, malgré les nombreuses études qui ont été menées (notamment du point de vue de la conception d'algorithmes *approchés*), il reste beaucoup de chemin à parcourir avant de parvenir à une connaissance satisfaisante des particularités de ces problèmes plus généraux.

Dans cette introduction, nous commençons par définir brièvement le cadre de l'*optimisation combinatoire* dans lequel s'inscrit cette thèse (section 1.1), puis nous donnons quelques définitions utilisées tout au long du document (section 1.2). Ensuite, nous définissons formellement les problèmes étudiés et détaillons les propriétés et résultats associés les plus importants (section 1.3). Enfin, nous concluons en donnant un aperçu de nos résultats et de l'organisation du présent document (section 1.4).

1.1 Rapide survol de l'optimisation combinatoire : notions fondamentales

Dans cette thèse, nous nous plaçons dans la problématique de l'optimisation combinatoire : trouver, à l'aide d'un algorithme *efficace* (ce concept est formalisé ci-dessous), une *meilleure* solution parmi un ensemble de solutions admissibles, fini mais très grand (classiquement, de taille exponentielle). Les applications industrielles de ce type de problématique sont nombreuses et variées, car beaucoup de problèmes peuvent se formuler de cette façon.

La *théorie de la complexité* vise à fournir un cadre formel pour un certain nombre de domaines de recherche, et en particulier pour l'optimisation combinatoire. La classe $\mathcal{NP}$ (pour *Non-determinist Polynomial*) est définie comme étant la classe des problèmes de décision que l'on peut résoudre à l'aide d'un algorithme non déterministe **polynomial**, c'est-à-dire dont la complexité est polynomiale en la taille du codage de l'instance. À l'intérieur de cette classe, la classe $\mathcal{P}$ est la classe des problèmes de décision que l'on peut résoudre à l'aide d'un algorithme *déterministe* polynomial. La conjecture la plus célèbre actuellement en informatique théorique est que $\mathcal{P} \neq \mathcal{NP}$; autrement dit, que certains problèmes de $\mathcal{NP}$ ne peuvent pas être résolus par des algorithmes (déterministes) efficaces. Pour toutes les notions que nous ne développerons (ou ne détaillerons) pas, le lecteur est invité à se reporter à l'excellente monographie de Michael Garey et David Johnson [76].

Un problème Π est dit **$\mathcal{NP}$-difficile** si l'existence d'un algorithme polynomial (déterministe) pour Π implique l'existence d'un algorithme polynomial (déterministe) pour n'importe quel problème de $\mathcal{NP}$. Un problème $\mathcal{NP}$-difficile est dit **$\mathcal{NP}$-complet** s'il appartient à $\mathcal{NP}$. Stephen A. Cook a, le premier, prouvé l'existence de tels problèmes, en montrant que le problème de satisfaction de contraintes SAT, issu de la logique, était $\mathcal{NP}$-complet [45]. Une façon de montrer la **complétude** dans $\mathcal{NP}$ d'un problème est d'utiliser la notion de **réduction polynomiale** [98] : étant donné un problème $\Pi \in \mathcal{NP}$ et un problème $\mathcal{NP}$-difficile Π^*, si Π^* se réduit polynomialement à Π (c'est-à-dire, en substance, s'il existe un algorithme polynomial qui transforme toute instance de Π^* en une instance de Π), alors Π est $\mathcal{NP}$-complet (ou $\mathcal{NP}$-difficile s'il n'est pas dans $\mathcal{NP}$). En règle générale, on montre qu'un problème d'*optimisation* est $\mathcal{NP}$-difficile en montrant que le problème de *décision* associé (qui consiste à décider s'il existe une solution admissible de valeur au moins ou au plus tant) est $\mathcal{NP}$-complet.

Comme de très nombreux problèmes étudiés dans la pratique sont $\mathcal{NP}$-difficiles (et donc, si $\mathcal{P} \neq \mathcal{NP}$, il n'existe pas d'algorithmes polynomiaux pour les résoudre), on est confronté à plusieurs alternatives pour les résoudre.

La première possibilité est de chercher à résoudre des cas particuliers d'un problème. Par exemple, on ajoute une hypothèse (réaliste, si possible) au problème, et on étudie l'impact de cette hypothèse sur la complexité du

problème (est-il toujours $\mathcal{NP}$-difficile, ou bien est-il à présent dans $\mathcal{P}$?). Pour les problèmes d'optimisation dans les graphes (comme ceux étudiés dans ce document), une hypothèse classique est de restreindre la structure du graphe. Cette approche est une de celles que nous utiliserons.

Une deuxième approche est celle de **l'approximation polynomiale** : on cherche alors à calculer, en temps polynomial, une solution qui n'est pas optimale mais *proche* de l'optimum (par opposition aux algorithmes **exacts**), cette proximité avec l'optimum pouvant être garantie *a priori*. Cette garantie d'approximation est contenue dans le **ratio d'approximation** : étant donné un problème d'optimisation Π et un réel $\alpha \geq 1$, un **algorithme α-approché** pour Π (c'est-à-dire un algorithme avec un ratio d'approximation α pour Π) est un algorithme polynomial $\mathcal{A}$ qui calcule une solution admissible pour Π telle que $\max_{I \ / \ I \text{ est une instance de } \Pi} \left\{ \frac{\mathrm{OPT}_I}{\mathrm{SOL}_{\mathcal{A}}(I)}, \frac{\mathrm{SOL}_{\mathcal{A}}(I)}{\mathrm{OPT}_I} \right\} \leq \alpha$, où OPT_I est la valeur optimale pour l'instance I du problème Π et $\mathrm{SOL}_{\mathcal{A}}(I)$ est la valeur de la solution donnée par $\mathcal{A}$ pour l'instance I du problème Π. Nous détaillons quelques aspects de l'approximation polynomiale en section 1.1.1.

Une troisième approche est de chercher des algorithmes (exacts), non pas polynomiaux, mais **pseudo-polynomiaux** ou bien **FPT** (pour *Fixed-Parameter Tractable*). Un algorithme est pseudo-polynomial s'il est polynomial en la taille des nombres qui interviennent dans la donnée de l'instance, et non nécessairement en la taille du codage de l'instance (coder un nombre n nécessitant normalement $\log n$ bits, et non n). Cependant, certains problèmes ne peuvent pas être résolus par des algorithmes pseudo-polynomiaux (sauf si $\mathcal{P} = \mathcal{NP}$) : de tels problèmes sont dits $\mathcal{NP}$-difficiles (ou $\mathcal{NP}$-complets s'ils sont dans $\mathcal{NP}$) **au sens fort**. Un exemple typique de problèmes $\mathcal{NP}$-difficiles au sens fort est la classe des problèmes $\mathcal{NP}$-difficiles qui sont **polynomialement bornés**, c'est-à-dire pour lesquels la taille des nombres intervenant dans la donnée des instances est bornée par un polynôme en la taille de leur codage (comme les problèmes Max SAT ou Max 2SAT, par exemple [76]). En particulier, cela signifie que la valeur optimale est bornée par un polynôme en la taille du codage.

Un algorithme FPT est un algorithme dont la complexité s'écrit $f(p)|\mathcal{I}|^{\alpha}$, où p est un *paramètre* du problème, f est une fonction continue de p, $|\mathcal{I}|$ est la taille du codage de l'instance $\mathcal{I}$ du problème à résoudre et $\alpha \geq 0$ est un réel fixé [56]. Une fois que le paramètre p a été identifié et fixé, l'algorithme devient donc polynomial et sa complexité est "raisonnable", au sens où l'exposant de $|\mathcal{I}|$ ne dépend pas de p. Un problème est FPT (ou dans FPT, car c'est aussi le nom de la classe) pour un certain paramètre p s'il admet un algorithme FPT pour ce paramètre. Il faut remarquer qu'un problème peut être FPT vis-à-vis d'un paramètre et ne pas l'être vis-à-vis d'un autre (le choix du paramètre est donc déterminant). Cette approche récente (elle a été introduite il y a 10 à 15 ans) a relancé l'espoir de trouver des algorithmes efficaces pour de nombreux problèmes $\mathcal{NP}$-difficiles pour lesquels on peut

identifier, dans les instances rencontrées en pratique, des paramètres naturels de faible valeur (par exemple, des problèmes de bio-informatique). Parfois, lorsqu'on ne peut prouver qu'un problème est FPT (ou lorsqu'on peut prouver qu'il n'est pas FPT moyennant une certaine conjecture considérée comme probable en théorie de la complexité), il peut aussi être intéressant de montrer que, lorsqu'un paramètre p du problème est fixé, il existe un algorithme polynomial pour le résoudre (même si la complexité de cet algorithme ne peut pas s'écrire sous la forme $f(p)|\mathcal{I}|^{\alpha}$, comme par exemple $|\mathcal{I}|^{p}$). Nous étudierons donc également cette approche.

Une quatrième approche consiste à chercher des méthodes de résolution exacte, non polynomiales en théorie, mais très efficaces en pratique (c'est-à-dire sur des instances réelles). Parmi les méthodes les plus utilisées on peut citer les méthodes par séparation et évaluation (*Branch and Bound* en anglais), avec éventuellement ajout de plans sécants (*Branch and Cut* en anglais). Le principe de ces méthodes est de construire un arbre de recherche (avec ajout de nouvelles contraintes à chaque nouveau nœud), et de résoudre en chaque nœud de cet arbre une **relaxation** du problème, calculable en temps polynomial (c'est l'étape du *calcul de borne*). En chaque nœud, la borne peut être obtenue de plusieurs façons : programmation linéaire (PL), programmation semi-définie (SDP), programmation quadratique convexe, etc. Nous détaillons certains aspects des méthodes par calcul de bornes en section 1.1.2.

Pour finir, une cinquième approche de résolution consiste à employer des **heuristiques**, c'est-à-dire des algorithmes, efficaces en pratique (et souvent polynomiaux en théorie), qui fournissent une solution approchée dont on ne sait pas garantir a priori la qualité (cette garantie peut parfois être obtenue *a posteriori*, en comparant la valeur de la solution fournie à une *borne*). On distingue les heuristiques et les méta-heuristiques. Les premières sont souvent associées à un problème particulier (ou à une classe de problèmes), pour lequel elles sont particulièrement adaptées : elles sont en général issues d'idées (ou de variantes d'idées) utilisées pour concevoir des algorithmes approchés (nous utiliserons cette démarche dans le chapitre 7). Les secondes sont des méthodes génériques de recherche de solutions (souvent par recherche locale), dont le schéma général demande à être adapté à chaque problème (on peut citer la méthode de recherche tabou, le recuit simulé, les algorithmes génétiques, les algorithmes par colonies de fourmis, etc.).

1.1.1 Une brève présentation de l'approximation polynomiale

La théorie de l'approximation polynomiale étant vaste [93, 159], nous nous contenterons uniquement d'en donner les bases, qui nous seront utiles dans la suite. L'un des points les plus intéressants est la *classification* des problèmes du point de vue de l'approximation, dans une classe ou dans une autre : cela permet d'affiner la notion de $\mathcal{NP}$-difficulté, certains problèmes $\mathcal{NP}$-difficiles se comportant très bien du point de vue de l'approximation,

d'autres au contraire très mal. Plusieurs classes d'approximation ont été définies, entraînant l'apparition de résultats d'**inapproximabilité** : en substance, cela signifie que, sauf si $\mathcal{P} = \mathcal{NP}$ (ou toute autre hypothèse peu plausible issue de la théorie de la complexité), tel problème ne peut pas appartenir à telle classe d'approximation, c'est-à-dire, il n'existe pas pour ce problème d'algorithme approché ayant tel ratio. Nous ne détaillerons pas tous les travaux fondamentaux qui ont permis d'établir une grande partie de ces résultats (définition de la classe MAX SNP [135], théorème PCP [9], etc.) ; nous allons simplement présenter les classes d'approximation que nous utiliserons le plus. Nous ne parlerons pas non plus de la notion d'**approximation différentielle**, car nous ne l'utiliserons pas [136].

On définit la classe ***APX*** comme étant la classe des problèmes d'optimisation admettant un algorithme α-approché pour un $\alpha \geq 1$ fixé (par exemple, le problème CouvertureParLesSommets admet un algorithme 2-approché, et appartient donc à APX). Un problème est ***APX*-difficile** s'il existe un réel $\epsilon > 0$ tel que, si $\mathcal{P} \neq \mathcal{NP}$, il n'existe pas d'algorithme $(1+\epsilon)$-approché pour ce problème (voir [9]). Un problème APX-difficile est ***APX*-complet** s'il appartient à APX. Étant donné un problème d'optimisation Π, on dit que Π admet un PTAS (pour *Polynomial-Time Approximation Scheme*), ou **schéma d'approximation polynomial** en français, s'il existe une famille d'algorithmes $(\mathcal{A}_\epsilon)_{\epsilon>0}$ telle que, pour tout $\epsilon > 0$ fixé, $\mathcal{A}_\epsilon$ est un algorithme $(1+\epsilon)$-approché pour Π. En d'autres termes, un problème APX-difficile n'admet pas de PTAS si $\mathcal{P} \neq \mathcal{NP}$ (et, inversement, un problème qui admet un PTAS n'est pas APX-difficile, sauf si $\mathcal{P} = \mathcal{NP}$). Une sous-classe des PTAS est la classe des FPTAS (pour *Fully Polynomial-Time Approximation Schemes*), **schémas d'approximation complètement polynomiaux** en français. Un FPTAS est simplement un PTAS dont le temps d'exécution peut s'écrire sous la forme : $O((\frac{1}{\epsilon})^\alpha |\mathcal{I}|^\beta)$, où ϵ est défini comme ci-dessus, $\alpha \geq 0$ et $\beta \geq 0$ sont deux réels fixés, et $|\mathcal{I}|$ est la taille du codage de l'instance considérée. Montrer qu'un problème n'admet pas de FPTAS (sauf si $\mathcal{P} = \mathcal{NP}$) est souvent beaucoup plus facile que de montrer qu'il n'admet pas de PTAS (sans compter que certains problèmes n'admettent effectivement pas de FPTAS, mais admettent un PTAS), grâce à la proposition classique suivante, dont la démonstration est laissée au lecteur en exercice :

Proposition 1.1. *Tout problème $\mathcal{NP}$-difficile (à valeurs entières) qui est polynomialement borné n'admet pas de FPTAS, sauf si $\mathcal{P} = \mathcal{NP}$.*

Un exemple typique est celui d'un problème où la donnée de l'instance ne nécessite la donnée d'aucun "grand" nombre (Max SAT, par exemple).

1.1.2 Les méthodes arborescentes et le calcul de bornes : introduction à la Programmation Linéaire

Une technique classique pour résoudre en pratique un problème $\mathcal{NP}$-difficile consiste à le modéliser par un **programme linéaire en nombres entiers** (PLNE), puis à résoudre ce PLNE à l'aide d'une des méthodes arborescentes décrites précédemment. L'avantage de cette technique est que de nombreux problèmes d'optimisation combinatoire peuvent se modéliser aisément et naturellement sous forme de PLNE : un **programme linéaire** (PL) consiste à optimiser une fonction linéaire sous des contraintes linéaires. Un PLNE est un PL où toutes les variables sont contraintes à être entières. Bien que la résolution d'un PLNE soit un problème $\mathcal{NP}$-difficile dans le cas général, il existe des algorithmes efficaces, en théorie ou en pratique, pour résoudre un PL *continu*, c'est-à-dire lorsque l'on autorise les variables à être réelles (méthodes de point intérieur, algorithme du simplexe, méthode des ellipsoïdes, etc.). Pour résoudre un PLNE, on effectue donc une recherche arborescente, et, en chaque nœud de l'arbre, on ajoute de nouvelles contraintes sur une variable (par exemple, si on a une variable $x_i \in \{0, 1\}$, on crée deux nœuds fils, l'un avec la contrainte $x_i = 0$, l'autre avec la contrainte $x_i = 1$), puis on calcule une borne en relâchant les contraintes d'intégrité sur les variables (**relaxation continue**) et en résolvant le PL continu ainsi obtenu.

Nous n'utiliserons pas de méthodes de résolution arborescentes dans la suite de ce document, mais nous utiliserons la modélisation par la PL pour certains problèmes. C'est pourquoi nous détaillons certains aspects ici.

Dualité. L'un des aspects les plus intéressants en PL est la **dualité** : à tout PL (P) de la forme $\max c \cdot x$ sous contraintes $Mx \leq b$ (c et b étant des vecteurs de données, M une matrice de données et $x \geq 0$ un vecteur d'inconnues), on peut associer un PL **dual** (D) de la forme $\min y \cdot b$ sous contraintes $yM \geq c$, $y \geq 0$. Le **théorème de la dualité forte** en PL affirme que (P) et (D) ont les mêmes valeurs optimales ; en outre, il existe des équations (appelées **relations d'exclusion** ou **conditions des écarts complémentaires**) liant les solutions optimales de ces deux PL [149].

Ratio et saut d'intégrité. Dans la majorité des cas, la valeur de la solution optimale (et donc entière) d'un PLNE donné n'est pas égale à la valeur optimale de sa relaxation continue (c'est-à-dire, du PL continu associé). On appelle **saut d'intégrité** la différence relative entre ces deux valeurs : il vaut $\frac{OPT_E - OPT_C}{OPT_E}$ pour un problème de minimisation et $\frac{OPT_C - OPT_E}{OPT_C}$ pour un problème de maximisation, où OPT_E et OPT_C désignent respectivement les valeurs optimales en entier et en continu (on suppose que $OPT_E > 0$ et $OPT_C > 0$). On appelle **ratio d'intégrité** la quantité $\max\left\{\frac{OPT_E}{OPT_C}, \frac{OPT_C}{OPT_E}\right\}$. Il faut noter que l'on peut essayer de construire une solution entière (ap-

prochée) à partir de la solution optimale continue (en l'arrondissant). Dans de nombreux cas, cependant, la meilleure borne que l'on sache obtenir en temps polynomial pour OPT_E est OPT_C. Cette technique fournit donc un algorithme approché dont le ratio d'approximation (si on cherche un minimum, par exemple) est borné par $\frac{SOL}{OPT_C}$ (où SOL est la valeur de la solution entière ainsi construite), ce qui est plus grand que le ratio d'intégrité : autrement dit, si l'on peut exhiber des instances où le ratio d'intégrité est "grand", alors on montre du même coup que cette approche ne pourra pas permettre de concevoir un "bon" algorithme approché.

Séparation. Un autre aspect important de la PL est l'équivalence entre *optimisation* et *séparation* : en substance, si, étant donné un **polyèdre** $Mx \leq b$, il existe un algorithme polynomial pour décider si, étant donné un point x^* de l'espace, x^* est à l'intérieur ou à l'extérieur de ce polyèdre (et, dans le second cas, pour exhiber une contrainte du polyèdre violée par x^*), alors il existe un algorithme polynomial (basé sur la **méthode des ellipsoïdes**) pour optimiser toute fonction linéaire sous les contraintes $Mx \leq b$. Ce résultat est fondamental lorsque l'on cherche à résoudre des PL dont le nombre de contraintes est exponentiel [85].

Totale unimodularité. Enfin, un autre aspect intéressant est un lien entre la PL et la PLNE. Sous certaines conditions, toute **solution de base** d'un PL (c'est-à-dire, toute solution qui sature un nombre de contraintes linéairement indépendantes égal à la dimension du polyèdre des contraintes ; nous y reviendrons ultérieurement) est à coordonnées entières. C'est ce qui se passe, par exemple, lorsque la matrice des contraintes M est **totalement unimodulaire** : dans ce cas, une solution optimale entière peut être obtenue en temps polynomial (et le saut d'intégrité est nul) [149].

1.2 Quelques notions de base en théorie des graphes

Les problèmes d'optimisation que nous étudierons se classent dans la famille des problèmes d'optimisation dans les graphes. Dans cette section, nous passons donc en revue l'ensemble des notions de théorie des graphes dont nous aurons besoin dans la suite de ce document (voir aussi [21]).

Notions générales. Étant donné un graphe $G = (S, A)$ orienté ou non orienté, S désignera l'ensemble de ses **sommets** et $A \subseteq S \times S$ l'ensemble de ses **arêtes** (ou de ses **arcs**, un arc étant une arête orientée, c'est-à-dire ordonnée). Étant donnée une arête $a = (u, v) = (v, u)$, les sommets u et v sont appelés **extrémités** de a. Étant donné un arc $a = (u, v)$, u est **l'extrémité initiale** de a, et v son **extrémité terminale**. Étant donné $E \subseteq A$, $G \setminus E$ est le graphe $G' = (S, A \setminus E)$, obtenu à partir de G en supprimant les arêtes

de E de l'ensemble A. Deux arêtes (resp. arcs) (u, v) et (v, w) sont dites **adjacentes** (resp. dits **adjacents**). Une **chaîne** est (une arête ou) une suite d'arêtes adjacentes, ou une suite d'arcs dont le graphe non orienté sous-jacent (c'est-à-dire, le graphe non orienté obtenu en supprimant l'orientation des arcs) est une chaîne. Un **chemin** est (un arc ou) une suite d'arcs adjacents. Un chemin (ou une chaîne) est dit **élémentaire** s'il ne contient pas deux fois le même sommet. Un **cycle** (resp. un **circuit**) est l'union d'une chaîne (resp. d'un chemin) élémentaire $u_1, \ldots, u_p$ et d'une arête (resp. d'un arc) (u_p, u_1).

Un graphe $G = (S, A)$ est **connexe** s'il existe une chaîne entre tout couple de sommets de S. Si un graphe n'est pas connexe, on appellera **composante connexe** chacun de ses sous-graphes connexes, maximal au sens de l'inclusion. Un graphe est dit **simple** s'il n'admet pas d'arêtes (ou d'arcs) parallèles (c'est-à-dire, identiques), et **sans boucles** s'il n'admet pas d'arêtes (ou d'arcs) de la forme (u, u), $u \in S$. Un graphe non orienté $G = (S, A)$ simple et sans boucles est dit **complet** si, pour toute paire de sommets u, v de S, $(u, v) \in A$. On notera K_n le graphe complet à n sommets. Un graphe $G = (S, A)$ est dit **biparti** ssi il existe un ensemble $U \subseteq S$ tel que, pour toute arête $(u, v) \in A$, $u \in U$ et $v \in S \setminus U$; G est **biparti complet** si, pour tout $u \in U$ et pour tout $v \in S \setminus U$, $(u, v) \in A$. On notera $K_{n_1, n-n_1}$ le graphe biparti complet avec n sommets et $|U| = n_1$.

Un graphe (ou un sous-graphe) $G = (S, A)$ est dit **2-sommet-connexe** (resp. **2-arête-connexe**) ssi pour tout couple de sommets de S il existe au moins deux chaînes reliant ces deux sommets et n'ayant aucun sommet (resp. aucune arête) en commun. Un **bloc** de G est un sous-graphe 2-sommet-connexe de G, maximal au sens de l'inclusion.

Étant donné un graphe non orienté $G = (S, A)$, le **degré** de $v \in S$ (dans G), noté $deg(v)$, est *le nombre d'arêtes adjacentes à* v :

$$deg(v) = |\{(v, u) \in A\}|$$

Pour un graphe orienté $G = (S, A)$ et un sommet $v \in S$, on définit le **demi-degré extérieur** de v, noté $deg^+(v)$, comme suit :

$$deg^+(v) = |\{(v, u) \in A\}|$$

Le **demi-degré intérieur** de v, noté $deg^-(v)$, est donné par :

$$deg^-(v) = |\{(u, v) \in A\}|$$

Pour un graphe orienté, le degré d'un sommet v est donc égal à :

$$deg(v) = deg^+(v) + deg^-(v)$$

On définit également, pour $v \in S$, $\Gamma^+(v) = \{u \in S \text{ tels que } (v, u) \in A\}$ et $\Gamma^-(v) = \{u \in S \text{ tels que } (u, v) \in A\}$; on a alors $|\Gamma^+(v)| = deg^+(v)$ et $|\Gamma^-(v)| = deg^-(v)$.

Un graphe simple $G = (S, A)$ est dit **pondéré** s'il existe une fonction $c : A \to \mathbb{N}^*$ (il est dit **non pondéré** sinon ; dans ce cas, on posera $c(a) = 1$ $\forall a \in A$). Étant donné un graphe simple $G = (S, A)$, pondéré et non orienté, le **degré pondéré** d'un sommet $v \in S$ est égal à :

$$deg_c(v) = \sum_{(u,v)\in A} c(u, v)$$

Pour un graphe orienté, le degré pondéré et les demi-degrés intérieur et extérieur pondérés sont définis de façon similaire. Un graphe connexe (pondéré) est dit **eulérien** si tous ses degrés (pondérés) sont pairs.

Étant donné un sous-ensemble $U \subseteq S$, on définit la **coupe** $\delta_G(U)$ comme l'ensemble des arêtes (u_i, u_j) dans A telles que $u_i \in U$ et $u_j \in S \setminus U$.

Arbres. Le **nombre cyclomatique** d'un graphe connexe non orienté $G = (S, A)$ est égal à $\nu(G) = |A| - |S| + 1$; c'est la dimension d'une base de cycles pour G. Un **arbre** est un graphe connexe sans cycle ; pour un arbre $G = (S, A)$ on a donc $\nu(G) = 0$ et $|A| = |S| - 1$. Une **feuille** est un sommet de degré 1 d'un arbre. Une **étoile** est un arbre dont tous les sommets, sauf un, sont des feuilles. Une **forêt** est un ensemble d'arbres. Une **arborescence** est un graphe orienté dont le graphe non orienté sous-jacent est un arbre, et qui possède un sommet unique, la **racine**, notée r, tel qu'il existe un (unique) chemin (orienté) de r vers n'importe quel autre sommet de l'arborescence. **Enraciner** un arbre signifie choisir un sommet v et orienter les arêtes de l'arbre de façon à obtenir une arborescence de racine v (dans ce cas, on dit aussi que l'arbre **prend racine** en v). Étant donnée une arborescence R de racine r, la **hauteur** d'un sommet v est sa distance (en nombre d'arcs) par rapport à r, c'est-à-dire la longueur du chemin de r à v. Un **palier** de R est un ensemble de sommets de même hauteur (maximal au sens de l'inclusion). La hauteur d'un palier est égale à la hauteur de ses sommets. La hauteur d'une arborescence est le maximum des hauteurs de ses paliers, c'est-à-dire la longueur d'un plus long chemin issu de la racine. Dans la suite, on utilisera la lettre T pour désigner des arbres (on pourra considérer, par exemple, que ce sont des *Tilleuls*).

Graphes planaires. Un graphe est dit **planaire** s'il peut être dessiné dans le plan sans que deux ou plusieurs de ses arêtes ne se croisent ailleurs qu'en des sommets ; une telle façon de dessiner un graphe planaire est appelée un **plongement** (*embedding* en anglais), et le graphe est alors dit **plongé dans le plan** (un graphe planaire admet une infinité de plongements différents).

Étant donné $G = (S, A)$ un graphe planaire plongé dans le plan, la **face extérieure** de G (*outer face* en anglais) est le contour délimitant le plongement de G. Une **face interne** (ou simplement **face**) de G est un cycle de G, minimal au sens de l'inclusion (un graphe planaire à ϕ faces internes a

donc au total $\phi + 1$ faces, en comptant la face extérieure). Deux faces sont **adjacentes** si elles ont au moins une arête en commun. Étant donné un bloc de G, le **cycle extérieur** de ce bloc est le cycle délimitant son contour. Le **cycle extérieur** de G est défini comme l'union disjointe des cycles extérieurs de tous ses blocs, et est donc inclus dans la face extérieure de G.

À tout graphe planaire $G = (S, A)$ on peut associer un graphe **dual** G^D planaire (pas nécessairement simple), ayant le même nombre d'arêtes. Les sommets de G^D sont les faces de G, et il y a une arête entre deux sommets de G^D ssi les deux faces correspondantes dans G sont adjacentes (il faut également noter qu'il y a une bijection entre les faces de G^D et les sommets de G). Une des propriétés fondamentales des graphes planaires est la **formule d'Euler** : étant donné un graphe connexe $G = (S, A)$, on a

$$\phi = |A| - |S| + 1$$

où ϕ est le nombre de faces internes de G. On peut remarquer que le nombre cyclomatique d'un graphe planaire à ϕ faces (internes) est donc ϕ (en fait, les ϕ faces forment une base de cycle). À l'aide de cette formule, on prouve aisément que

$$|A| \leq 3|S| - 6$$

et que

$$\phi + 1 \leq 2|S| - 4 \tag{1.1}$$

Étant donné un entier $k \geq 1$, un graphe simple est dit **planaire à k niveaux de sommets** (*k-outerplanar* en anglais), ou simplement **planaire à k niveaux**, si c'est un graphe planaire qui admet un plongement ayant au plus k **niveaux** de sommets, c'est-à-dire tel que, après avoir supprimé de façon itérative les sommets se trouvant sur la face extérieure au plus k fois, on obtient le graphe vide [12]. Il faut noter que tout graphe planaire est planaire à k niveaux pour un certain k, et que la classe des graphes planaires à k niveaux pour k fixé (dans toute la suite du document, on supposera systématiquement que k est fixé lorsque l'on considérera des graphes planaires à k niveaux) est bien connue pour être une sous-classe importante des graphes planaires de largeur d'arbre bornée [23] (voir plus bas). Par exemple, une **grille rectangulaire** (voir [69] pour une définition formelle) à N lignes et $2M$ colonnes ($N > 2M$) est un graphe planaire à M niveaux.

En particulier, un graphe **planaire superficiel** (*outerplanar* en anglais) est un graphe planaire contenant au moins un sommet et admettant un plongement où tous ses sommets sont sur la face extérieure ; en d'autres termes, c'est un graphe planaire à 1 niveau.

Un **cactus** est un graphe (simple) tel qu'aucune de ses arêtes n'appartient à plus d'un cycle. Un cactus **triangulaire** ne contient aucun cycle de longueur 4 ou plus. On peut facilement montrer que les cactus sont en fait les graphes planaires dont toutes les arêtes sont sur la face extérieure ; ils forment

donc une sous-classe des graphes planaires superficiels. Une sous-classe importante des cactus est la classe des **arbres d'anneaux** (*trees of rings* en anglais [61]) ; un arbre d'anneaux est en fait un cactus 2-arête-connexe (un **anneau** étant un graphe composé d'un seul cycle).

Largeur d'arbre. Définissons à présent la notion de **largeur d'arbre** pour les graphes orientés et non orientés. Il faut noter que cette notion (ou "métrique", puisque, fondamentalement, c'est une certaine mesure de la distance séparant un graphe donné de la structure d'un arbre) s'est révélée très importante au cours de ces dernières années : beaucoup de problèmes difficiles (au sens de la complexité) se sont révélés faciles (toujours au sens de la complexité) dans les graphes où ce paramètre est borné par une constante.

Commençons par définir la largeur d'arbre d'un graphe non orienté [145] :

Définition 1.1. *Soit $G = (S, A)$ un graphe non orienté. Un couple $\Psi = (T, (X_w)_{w \text{ est un sommet de } T})$, consistant en un arbre T et en un multi-ensemble dont les éléments $X_w \subseteq S$ (appelés* **sacs***) sont indicés par les sommets de T, est une* **décomposition arborescente** *de G s'il vérifie :*

1. *$\forall (u, v) \in A, \exists w$ sommet de T tel que $u \in X_w$ et $v \in X_w$;*
2. *pour chaque $v \in S$, les sommets w tels que $v \in X_w$ induisent un sous-graphe connexe de T.*

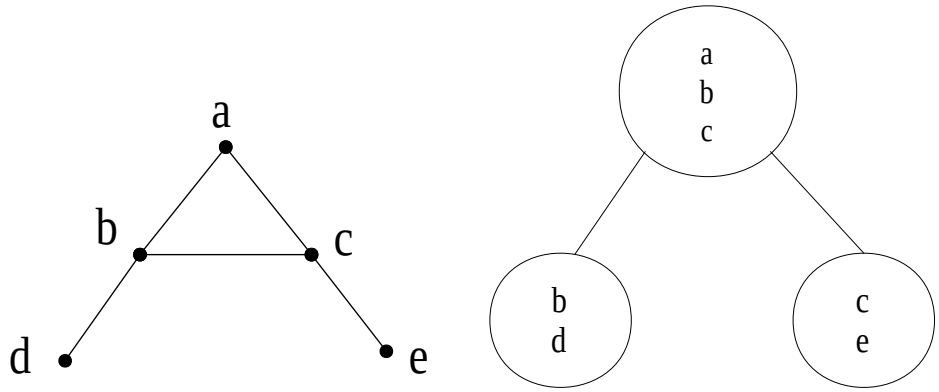
On définit la **largeur** *d'une décomposition arborescente Ψ comme étant égale à $\max_{w \text{ est un sommet de } T} |X_w| - 1$ et la* **largeur d'arbre** *de G, notée $la(G)$, est le minimum des largeurs de toutes les décompositions arborescentes de G. La* **largeur de chaîne** *d'un graphe G est la largeur minimum d'une décomposition arborescente pour G où l'arbre T est une chaîne.*

La figure 1.1 donne un exemple de décomposition arborescente pour un graphe non orienté.

Il faut remarquer, en particulier, que les arbres ont une largeur d'arbre de 1 (et, plus généralement, les graphes de nombre cyclomatique γ ont une largeur d'arbre $\Theta(\gamma)$), et les graphes planaires à k niveaux une largeur d'arbre $\Theta(k)$ [23]. Pour les graphes orientés, on utilise la définition donnée dans [96] :

Définition 1.2. *Étant donnée une arborescence R, si r, r' sont deux sommets de R, on écrit $r' > r$ si $r' \neq r$ et s'il existe un chemin orienté de r à r' dans R. Si a est un arc de R, on écrit $r' > a$ si $r' = r$ ou $r' > r$, où r' est un sommet de R et r est l'extrémité initiale de a. On écrit aussi $a \sim r$ quand l'arc a est incident à un sommet r.*

*Soit $D = (S, A)$ un graphe orienté et soit $U \subseteq S$. Le graphe orienté obtenu à partir de D en supprimant U sera désigné par $D - U$. On dira qu'un ensemble $Q \subseteq S$ est U-**normal** s'il n'existe aucun chemin orienté (élémentaire ou non) dans $D - U$ ayant ses sommets initial et final dans Q qui contient un sommet de $D - (U \cup Q)$. En d'autres termes, tout chemin*



(a) Un graphe non orienté de largeur d'arbre (et de chaîne) 2.

(b) Une décomposition arborescente de largeur 2 associée.

FIG. 1.1 – Un exemple de décomposition arborescente de largeur 2.

orienté (élémentaire ou non) dans $D - U$ qui commence et finit dans Q est entièrement contenu dans Q.

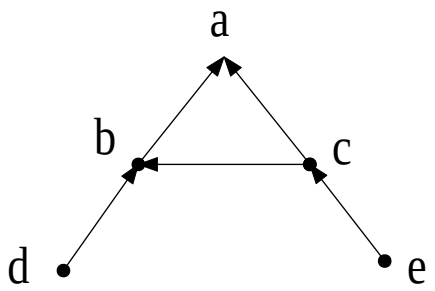
Une **décomposition arborescente orientée** *d'un graphe orienté simple $D = (S, A)$ est un triplet (R, X, W), où $R = (U, F)$ est une arborescence, et où $X = (X_a \subseteq S : a \in F)$ et $W = (W_r \subseteq S : r \in U)$ vérifient :*

1. *$(W_r : r \in U)$ est une partition de S ;*
2. *si $a \in F$, alors $\bigcup\{W_r : r \in U, r > a\}$ est X_a-normal.*

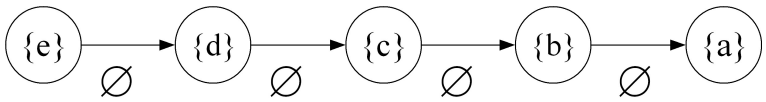
On définit la **largeur** *de (R, X, W) comme le plus petit entier w tel que $|(\bigcup_{a \sim r} X_a) \cup W_r| \leq w + 1$ pour chaque r dans U. La* **largeur d'arbre orientée** *de D est le plus petit entier w tel que D admet une décomposition arborescente de largeur w.*

La figure 1.2 donne un exemple de décomposition arborescente orientée.

Il faut noter que la notion de largeur d'arbre orientée diffère de la notion de largeur d'arbre non orientée, puisqu'un graphe orienté peut être de largeur d'arbre bornée alors que le graphe non orienté sous-jacent ne l'est pas. En particulier, les graphes de largeur d'arbre orientée égale à 0 sont les graphes orientés sans circuits (ou **GSC** ; en anglais, on parle de *Directed Acyclic Graphs*, ou DAG). Néanmoins, étant donné un graphe non orienté $G = (S, A)$, le graphe orienté obtenu à partir de G en remplaçant chaque arête (u, v) de A par deux arcs opposés (u, v) et (v, u) (on parle alors de **graphe bi-orienté** ou **symétrique**) a la même largeur d'arbre (orientée) que G [96].



(a) Un graphe orienté de largeur d'arbre 0.



(b) Une décomposition arborescente orientée de largeur 0 associée.

FIG. 1.2 – Un exemple de décomposition arborescente orientée de largeur 0.

1.3 Problèmes de multiflot entier et de multicoupe : introduction et état de l'art

Dans cette thèse, nous focalisons notre attention sur des problèmes de multiflot entier et de multicoupe. Dans cette section, nous introduisons ces problèmes et donnons les résultats fondamentaux qui y sont associés.

1.3.1 Définitions

Dans la suite du document, étant donné un problème ou un programme linéaire Π, on notera $Opt(\Pi)$ sa valeur optimale.

Soit $G = (S, A)$ un graphe orienté ou non, ayant $n = |S|$ sommets et $m = |A|$ arêtes (ou arcs). Soit c une fonction de pondération, $c : A \to \mathbb{Z}^+$, et soit $\mathcal{L}$ une liste de paires (source s_i, puits s'_i) de sommets de S (avec $s_i \neq s'_i$ $\forall i$). Ces paires sont appelées **liaisons**, les s_i et les s'_i sont appelés **terminaux**, et tout sommet extrémité d'une liaison est appelé **sommet terminal**. G est appelé le **graphe support**, et on appellera **graphe de demandes** le graphe H dont les sommets sont les sommets terminaux et dont les arêtes sont les liaisons. On notera $G + H$ le graphe obtenu à partir de G en ajoutant les arêtes (s_i, s'_i), $i \in \{1, \ldots, |\mathcal{L}|\}$. Pour chaque $i \in \{1, \ldots, |\mathcal{L}|\}$, on désignera par P_i l'ensemble des chaînes (ou chemins) élémentaires reliant s_i à s'_i dans G. En outre, on notera $P = \bigcup_{i \in \{1,\ldots,|\mathcal{L}|\}} P_i$.

Définissons à présent les problèmes que nous aurons à considérer (chaque problème existe en version orientée et non orientée, selon que le graphe considéré est orienté ou non). Tout d'abord, nous présentons les deux problèmes

qui nous intéressent plus particulièrement : MCM et MFEM.

Définition 1.3. Problème du multiflot entier maximum (désigné par **MFEM**) : *l'objectif est de router le nombre maximum d'unités de flot, chaque unité étant routée de s_i vers s'_i pour un i (l'ensemble de ces unités définissant un* **multiflot***), de façon à ce que, pour chaque arête ou arc a, le nombre* total *d'unités de flot qui la traversent soit inférieur ou égal à $c(a)$* **(contraintes de capacité)**.

Lorsque nous traitons des problèmes de flot (comme MFEM), les $c(a)$ seront appelés indifféremment **poids** ou **capacités**. Précisons également que, dans nos figures, nous utiliserons la notation "[c]" pour désigner une capacité (ou un poids) de valeur c.

Définition 1.4. Problème de la multicoupe minimum (noté **MCM**) : *l'objectif est de sélectionner un ensemble d'arêtes (ou d'arcs) de poids minimum (le poids d'un arête ou d'un arc a étant $c(a)$) dont la suppression ne laisse aucun chemin entre s_i et s'_i pour tout i (l'ensemble de ces arêtes définit une* **multicoupe***, et chaque liaison est alors dite* **coupée** *ou* **déconnectée***).*

Nous aurons également besoin d'un certain nombre de leurs variantes, que nous présentons à présent. Dans la suite du document, étant donnés une multicoupe C et un multiflot F, on notera $\|C\|$ et $\|F\|$ leurs *valeurs* respectives. En outre, un **chemin de routage** dans F sera un chemin (ou une chaîne) acheminant au moins une unité de flot associée à n'importe quelle liaison. On peut aussi remarquer que MFEM et MCM sont polynomialement bornés si la fonction de pondération c est bornée par un polynôme en m.

Problème du flot multiterminal entier maximum (noté FMTEM). C'est un cas particulier de MFEM où, étant donné un ensemble de sommets $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$, les liaisons sont (t_i, t_j) pour $i \neq j$.

Problème de la coupe multiterminale minimum (noté CMTM). C'est un cas particulier de MCM où, étant donné un ensemble de sommets $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$, les liaisons sont (t_i, t_j) pour $i \neq j$.

Problème de la maximisation du nombre de chemins disjoints par les arêtes (noté MCDA). C'est une variante de MFEM où $c(a) = 1$ pour toute arête ou tout arc a, et cela revient à sélectionner dans P un sous-ensemble de chemins disjoints par les arêtes de cardinalité maximum.

Problème de la maximisation du nombre de chemins disjoints par les sommets (noté MCDS). C'est une variante de MCDA (l'objectif est le même) où l'on cherche à sélectionner des chemins disjoints par les sommets.

Problème du multiflot entier maximum insécable (noté MFEMI). C'est une variante de MFEM où, pour tout i, les unités de flot routées de s_i à s'_i doivent transiter par le *même* chemin.

Problème de la maximisation du nombre de liaisons routées suivant des chemins disjoints par les arêtes (noté MLCDA). L'objectif est de maximiser le nombre de liaisons (s_i, s'_i) reliées par des chemins disjoints (un chemin par liaison, contrairement à MCDA) ; c'est une variante de MFEMI où $c(a) = 1$ pour toute arête ou tout arc a. (Toute solution admissible pour MLCDA est une solution admissible pour MCDA, et on a donc $Opt(\text{MLCDA}) \leq Opt(\text{MCDA})$.)

Problème de la multicoupe minimum sur les sommets (désigné par MCMS). C'est une variante de MCM (l'objectif est le même) où l'on supprime des sommets au lieu de supprimer des arêtes ou des arcs.

Problème de la coupe multiterminale minimum sur les sommets (noté CMTMS). C'est une variante de CMTM (l'objectif est le même) où l'on enlève des sommets au lieu d'enlever des arêtes ou des arcs.

Problème du multiflot entier avec demandes (noté MFED). C'est une version de décision du problème de multiflot entier, qui, étant données des demandes d_i associées aux liaisons (s_i, s'_i) (pour chaque i, d_i représente le nombre d'unités de flot qui doivent être routées entre s_i et s'_i), requiert de satisfaire ces demandes tout en respectant les contraintes de capacités ; une variante de ce problème est de trouver une solution admissible, c'est-à-dire respectant les demandes et les contraintes de capacités, qui minimise un coût (voir par exemple le problème MCLM en section 5.1).

Problème du multiflot continu avec demandes (noté MFCD). C'est une version fractionnaire de MFED ; en d'autres termes, on autorise les flots à ne pas être entiers.

Problème de l'existence de chemins disjoints par les arêtes (noté CDA). L'objectif est de décider si toutes les liaisons peuvent être reliées par des chemins disjoints (un chemin par liaison) ; c'est un problème de décision associé à MLCDA.

CDA est l'un des problèmes fondamentaux montrés $\mathcal{NP}$-complets par Richard Karp dans son célèbre article de 1972 [98]. En outre, il n'est pas difficile de voir que CDA est polynomialement borné.

Problème de l'existence de chemins disjoints par les sommets (noté CDS). C'est une variante de CDA (l'objectif est le même) où l'on cherche à construire des chemins disjoints par les sommets.

On peut remarquer que, pour tous ces problèmes, on peut supposer sans perte de généralité que le graphe support G est connexe (car, si ce n'est pas le cas, on peut traiter chaque composante connexe indépendamment).

Nous donnons à présent la définition de trois (autres) variantes des problèmes précédents, sur lesquelles nous ne nous pencherons pas dans cette thèse, mais qui ont également été beaucoup étudiées dans la littérature.

Problème du multiflot entier avec le maximum de demandes (noté MFEMD). C'est un problème d'optimisation associé à MFED, différent de MFEM. Étant données des demandes d_i et des poids w_i associés aux liaisons (s_i, s'_i) (pour chaque i, d_i est le nombre d'unités de flot qui doivent être routées entre s_i et s'_i), MFEMD requiert de satisfaire un sous-ensemble de demandes de poids maximum tout en respectant les contraintes de capacité (chaque demande sélectionnée devant être satisfaite intégralement).

Une autre variante très étudiée de ce problème consiste à router chaque demande sélectionnée suivant un seul chemin. Enfin, une dernière variante consiste à router les demandes sélectionnées, sans contrainte d'intégrité sur les flots (mais chaque demande sélectionnée doit toujours être routée en totalité) : c'est le problème de multiflot *tout-ou-rien* (*all-or-nothing* en anglais).

Problème du multiflot entier concurrent maximum (désigné par MFECM). C'est une variante de MFEMD où l'on cherche à maximiser le réel $0 \leq \alpha \leq 1$ tel que l'on puisse simultanément router un pourcentage α de chaque demande.

Problème de la K-coupe (noté KC). Il consiste à partitionner les sommet du graphe en K paquets non vides, de façon à minimiser la somme des poids des arêtes se trouvant entre ces paquets (c'est-à-dire, ayant leurs deux extrémités dans deux paquets différents). Pour K fixé, c'est un cas particulier de CMTM.

On peut trouver les principaux résultats concernant ces problèmes dans [13, 29, 33, 34, 77, 89, 103, 106, 108, 109, 156] (pour MFEMD et ses variantes), dans [11, 77, 103, 117] (pour MFECM) et dans [42, 83, 122, 164] (pour KC), ainsi que dans [93, Chapitre *Cut problems and their application to divide-and-conquer* par D.B. Shmoys (pp. 192–235)] (pour MFECM et KC). Nous ne parlerons pas non plus des problèmes de routage où l'on cherche à minimiser la "charge" des arêtes ou des arcs (c'est-à-dire, en substance, leur capacité idéale), et qui diffèrent généralement, dans leur traitement et

leur résolution, des problèmes avec poids fixes (voir [7] par exemple). Nous ne nous intéresserons pas non plus aux méthodes de résolution (exactes ou approchées) des problèmes en variables continues (et notamment le multiflot continu maximum ou le multiflot à coût minimal avec demandes) : quelques références concernant ces problèmes peuvent être trouvées dans [120, 165].

1.3.2 Motivations et applications

MFEM et MCM (ainsi que l'ensemble de leurs variantes) couvrent une quantité importante d'applications. Les plus évidentes concernent bien sûr les problématiques de routage (maximisation du débit ou du revenu) ou de fiabilité dans des réseaux de télécommunications.

Par ailleurs, CDA a été beaucoup étudié du fait de son application dans la conception de circuits VLSI [112, 118, 129]. En outre, des problèmes de coloration de chemins se posent dans les réseaux optiques [3, 139, 141] : une communication est un chemin entre deux nœuds du réseau, chaque communication occupe une longueur d'onde donnée, et l'objectif est de passer toutes les communications en utilisant un nombre minimum de longueurs d'ondes (couleurs). C'est en fait un problème de coloration de chemins, où les chemins de même couleur doivent être disjoints par les arêtes, et où l'on cherche à minimiser le nombre de couleurs utilisées. Bien que ce problème ne soit pas directement un de ceux étudiés dans cette thèse, il est facile de voir que des algorithmes performants pour MCDA peuvent servir de base à une résolution heuristique efficace de problèmes de coloration de chemins ; en fait, il a même été montré dans [10] qu'un algorithme α-approché pour MCDA pouvait être converti en un algorithme $O(\alpha \log n)$-approché pour la coloration de chemins.

Une liste des applications possibles du multiflot entier (à coût minimum) avec demandes peut être trouvée dans [48, 49, 124, 165] ; on peut citer, par exemple, le câblage d'usines EDF ou l'ordonnancement de véhicules dans les transports publics. Brunetta et al. [25] citent également une application de MFEM à la téléphonie : il s'agit d'un problème de routage d'appels dans un réseau téléphonique en période de forte demande.

Par ailleurs, Shmoys [93, Chapitre *Cut problems and their application to divide-and-conquer*, pp. 192–235] donne un cadre applicatif général pour MCM et ses variantes : de nombreux problèmes combinatoires de graphe peuvent être résolus, de façon exacte ou heuristique, par des approches de type *diviser-pour-régner* (*divide-and-conquer* en anglais). Or, ces approches reposent sur un partitionnement judicieux du graphe, qui se ramène souvent à des problèmes comme MCM ou CMTM. On peut aussi citer des problèmes de reconnaissance des communautés virtuelles d'internet, de partitionnement dans des réseaux sociaux (ou assimilés, voir [127] par exemple) ou dans des graphes de dépendances de type processeurs-tâches pour des problèmes de placement de tâches. Nous donnons à présent une brève illustration de ce

type d'applications.

Considérons le problème de placement de tâches suivant : plusieurs tâches sont à exécuter et plusieurs processeurs sont disponibles. Chaque tâche j_i a (ou pas) un processeur *préféré* π_h, sur lequel son coût d'exécution e_i est faible (disons 0, en prétraitant les données correctement). Sur tous les autres processeurs, le coût d'exécution de j_i est $e'_i > e_i$ (on prendra $e'_i := e'_i - e_i$ pour simplifier). Si j_i n'a pas de processeur préféré, son coût d'exécution sera le même sur tous les processeurs (et on le prendra donc égal à 0). Enfin, les tâches doivent communiquer entre elles : le coût de communication entre deux tâches j_i et j_l est c_{il}, et il s'applique si ces deux tâches ne sont pas exécutées sur le même processeur. L'objectif est d'exécuter (de placer) les tâches sur les processeurs de façon à minimiser la somme des coûts d'exécution et de communication.

Une instance de ce problème peut se modéliser comme une instance de CMTM : on définit un sommet pour chaque tâche et pour chaque processeur, puis on relie chaque paire de tâches (sommets) j_i et j_l par une arête de poids c_{il}, et chaque tâche j_i à un processeur π_h par une arête de poids e'_i ssi π_h est le processeur préféré de j_i. Alors, il n'est pas difficile de montrer qu'un placement optimal des tâches sur les processeurs correspond à une coupe multiterminale minimum dans ce graphe, où les terminaux sont les sommets représentant les processeurs (une tâche j_i étant considérée placée sur un processeur π_h si, une fois les arêtes de la solution optimale retirées, il existe une chaîne reliant le sommet j_i au sommet π_h).

1.3.3 Résultats antérieurs à cette thèse

Réductions

Il existe en fait des relations intéressantes entre les problèmes définis précédemment : certains peuvent se réduire à d'autres, c'est-à-dire que toute instance d'un certain problème peut se modéliser comme une instance d'un autre problème. Nous donnons la liste des réductions dans cette section (voir aussi [151, pp. 1223-1224]) : elles établissent une relation d'ordre sur la complexité et l'approximabilité de ces problèmes (les problèmes se réduisant à d'autres étant "les plus faciles").

CDA se réduit à CDS (dans les graphes orientés ou non), en remplaçant le graphe par son **graphe des arêtes** (*line graph* en anglais) : dans ce nouveau graphe, on associe un sommet à chaque arête (ou arc) du graphe initial, et on relie deux sommets si et seulement si, dans le graphe initial, les deux arêtes (ou arcs) associées ont une extrémité en commun.

De surcroît, dans les graphes orientés, CDS se réduit à CDA en appliquant, en chaque sommet, la transformation donnée dans la figure 1.3.

On peut remarquer que les réductions présentées ne préservent pas la planarité. Néanmoins, on peut réduire CDS dans les graphes non orientés

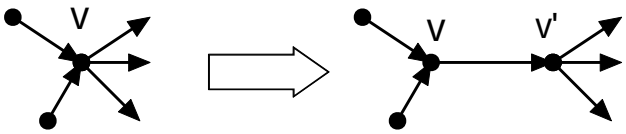


FIG. 1.3 – Réduire CDS à CDA dans les graphes orientés.

à CDS dans les graphes orientés à l'aide d'une réduction qui préserve la planarité : il suffit de remplacer chaque arête par une paire d'arcs opposés (ce qui revient à transformer le graphe non orienté en graphe bi-orienté). Enfin, on peut réduire CDA dans les graphes non orientés à CDA dans les graphes orientés à l'aide de la construction suivante (il faut remplacer chaque arête (u, v) par le "gadget" dessiné en figure 1.4).

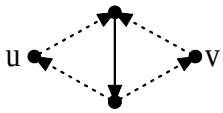


FIG. 1.4 – Réduction du cas non orienté au cas orienté pour CDA.

Cette transformation préserve également la planarité. Si l'arête (u, v) est pondérée, alors les cinq arcs ont le même poids qu'elle. Notons que toutes les réductions présentées s'appliquent également à MFEM, MCDA/MCDS, MCM/MCMS et CMTM/CMTMS.

Dualité flot-coupe et modélisations PL

Il faut noter que, lorsque $|\mathcal{L}| = 1$, MFEM et MCM sont respectivement équivalents au **problème du flot maximum** et au **problème de la coupe minimum**. Ces deux problèmes sont des problèmes classiques en optimisation combinatoire et en théorie des graphes, et ils ont été beaucoup étudiés ; en particulier, ce sont deux problèmes duaux, au sens de la PL. En d'autres termes, chacun des deux est modélisable par un PLNE, et les relaxations continues des deux PLNE sont duales l'une de l'autre. En outre, on peut montrer que les matrices de contraintes des deux relaxations continues sont totalement unimodulaires (**TU**) [5].

Ceci a pour première conséquence que tout flot maximum (entier) a la même valeur que toute coupe minimum : c'est le fameux **théorème de Ford-Fulkerson** [65], qui peut également se démontrer à l'aide d'arguments de théorie des graphes.

Une deuxième conséquence de la totale unimodularité des matrices de contraintes est que les deux problèmes peuvent être résolus efficacement ;

en d'autres termes, ils sont dans $\mathcal{P}$. En réalité, de nombreux algorithmes combinatoires efficaces ont été conçus pour résoudre ces deux problèmes sans passer par la PL [5].

Malheureusement, ces propriétés ne se conservent pas pour $|\mathcal{L}| > 1$: en règle générale, les valeurs optimales de MFEM et MCM ne sont pas égales, et les matrices associées aux modélisations en PL ne sont pas TU. Pour $|\mathcal{L}| = 2$, il a cependant été montré que, dans les graphes non orientés eulériens (pondérés ou non), la valeur de tout multiflot maximum (entier) est égale à la valeur de toute multicoupe minimum, et les deux problèmes sont polynomiaux dans ce cas [94, 142].

Quoi qu'il en soit, même s'il n'existe pas de relation liant les valeurs optimales entières, il existe néanmoins une relation intéressante entre MFEM et MCM. Ces deux problèmes peuvent se modéliser par des PLNE dont les relaxations continues sont duales (au sens de la PL) ; en d'autres termes, les valeurs optimales sont égales *en continu* [79]. Nous donnons à présent les formulations PL de MFEM et MCM. Il faut noter, à ce propos, qu'il existe deux façons de modéliser chacun des deux problèmes par la PL.

La première formulation est appelée "sommet-arc". Nous la donnons pour les graphes orientés, puisque le cas non orienté s'y réduit (voir la figure 1.4). Pour chaque arc (u, v) du graphe et pour chaque liaison (s_i, s'_i), on définit $f^i_{u,v}$ comme la valeur du i^e flot sur l'arc (u, v). Comme pour le flot maximum simple [5], on modélise alors MFEM à l'aide des contraintes de capacité et des contraintes de conservation du flot en chaque sommet :

$$
\textbf{(IPL-MFEM-1)} \left|
\begin{array}{lll}
\max & \displaystyle\sum_{i=1}^{|\mathcal{L}|} \left(\sum_{(s_i,u)\in A} f^i_{s_i,u} - \sum_{(u,s_i)\in A} f^i_{u,s_i} \right) & \\
\text{s. c.} & \displaystyle\sum_{(u,v)\in A} f^i_{u,v} = \sum_{(v,w)\in A} f^i_{v,w} & \forall i \in \{1,\ldots,|\mathcal{L}|\}, \forall v \in S \setminus \{s_i, s'_i\} \\
& \displaystyle\sum_{i=1}^{|\mathcal{L}|} f^i_{u,v} \leq c(u,v) & \forall (u,v) \in A \\
& f^i_{u,v} \in \mathbb{N} & \forall i \in \{1,\ldots,|\mathcal{L}|\}, \forall (u,v) \in A
\end{array}
\right.
$$

En dualisant ce PL, on obtient une formulation en PL pour MCM (voir [79]). Ces deux formulations ont l'avantage d'avoir un nombre de variables et de contraintes polynomial en la taille du graphe. Néanmoins, nous ne nous référerons pratiquement jamais à ces formulations, aussi ne les détaillerons-nous pas davantage ; nous proposons une formulation adaptée au cas non orienté dans le chapitre 7. Nous donnons à présent les deux formulations que nous utiliserons le plus pour MCM et MFEM. Elles présentent l'inconvénient d'être de taille exponentielle en la taille du graphe, mais rendent la relation de dualité entre les deux problèmes évidente.

Voici tout d'abord une formulation alternative pour MFEM (dite "arc-

chemin"). Pour tout chemin $p_i \in P$, on note f_i la quantité de flot qui circule sur p_i. On a alors :

$$(\textbf{\textit{IPL-MFEM-2}}) \left| \begin{array}{lll} \max & \sum_{i=1}^{|P|} f_i & \\ \text{s. c.} & \sum_{i \text{ t. q. } a \in p_i} f_i \leq c(a) & \forall a \in A \\ & f_i \in \mathbb{N} & \forall i \in \{1, \ldots, |P|\} \end{array} \right. \tag{1.2}$$

En associant une variable duale z_a à chaque contrainte du type (1.2) (contrainte de capacité), et en relâchant les contraintes d'intégrité sur les f_i (on les remplace par $f_i \geq 0, \forall i$), on obtient le dual suivant :

$$(\textbf{\textit{IPL-MCM-2}}) \left| \begin{array}{lll} \min & \sum_{a \in A} c(a) z_a & \\ \text{s. c.} & \sum_{a \in p_i} z_a \geq 1 & \forall i \in \{1, \ldots, |P|\} \\ & z_a \geq 0 & \forall a \in A \end{array} \right. \tag{1.3}$$

Il n'est pas difficile de voir que, pour tout arc $a \in A$, on a $z_a \leq 1$ dans toute solution optimale (à cause de la forme particulière de la fonction objectif). On peut donc ajouter la contrainte $z_a \in [0, 1], \forall a \in A$, sans changer le problème. Si on ajoute des contraintes d'intégrité sur les variables, on obtient : $z_a \in \{0, 1\}, \forall a \in A$. On a alors un PLNE modélisant MCM : en effet, la variable z_a est égale à 1 ssi l'arc a est sélectionné, et les contraintes (1.3) garantissent que, sur tout chemin reliant une source à son puits, au moins un arc sera sélectionné.

La relation de dualité entre les PL continus a pour conséquence que la valeur de tout multiflot admissible est inférieure à la valeur de toute multicoupe ; cette propriété importante a été utilisée pour concevoir des algorithmes approchés pour MFEM et MCM. Par exemple, si on a un algorithme polynomial qui calcule un multiflot entier F et une multicoupe C tels que $\|C\| \leq \alpha \|F\|$, alors on a

$$\|C^*\| \leq \|C\| \leq \alpha \|F\| \leq \alpha \|C^*\|$$

et

$$\|F\| \leq \|F^*\| \leq \|C\| \leq \alpha \|F\|$$

où F^* et C^* sont des solutions optimales pour MFEM et MCM respectivement. Autrement dit, cet algorithme est un algorithme α-approché à la fois pour MFEM et pour MCM. Cette approche, qui consiste à construire une solution entière pour le primal et une solution entière pour le dual en se basant

sur les relations d'exclusion, est appelée *approche primale-duale* (voir [93, Chapitre *The primal-dual method for approximation algorithms and its application to network design problems* par M.X. Goemans et D.P. Williamson (pp. 144-191)] pour une présentation). Remarquons cependant que le ratio d'approximation d'un algorithme de ce type ne peut jamais être meilleur que le ratio d'intégrité pour MCM et MFEM (qui prend alors toute son importance), à cause de la relation de dualité qui lie ces deux problèmes. Il est à noter qu'un algorithme 2-approché pour MCM et MFEM dans les arbres utilise cette approche [80] : le simple exemple d'une instance de FMTEM et CMTM dans une étoile non pondérée à trois feuilles, où les terminaux sont les feuilles, montre que le ratio entre les solutions optimales de ces deux problèmes peut effectivement être égal à 2 (le multiflot entier maximum valant 1 et la multicoupe minimum valant 2). Néanmoins, on peut remarquer que, dans cette instance, le ratio d'intégrité pour MFEM n'est égal qu'à 1.5 (le multiflot fractionnaire maximum valant 1.5) ; en fait, on ne connaît pas d'exemple d'arbre où ce ratio est plus grand que 1.5. En outre, bien que le ratio d'intégrité pour MCM soit 4/3 dans cette instance, on peut facilement construire des instances où il est arbitrairement proche de 2 : si on considère une instance de CMTM dans une étoile non pondérée à n feuilles, où les terminaux sont les feuilles, on obtient un ratio d'intégrité de $\frac{2(n-1)}{n}$, qui tend vers 2 quand n augmente. Il nous reste un dernière remarque à faire concernant la méthode primale-duale : elle peut aussi servir à calculer des solutions optimales entières lorsque la matrice des contraintes est TU [50].

Comme les ratios d'intégrité jouent un rôle important pour l'approximation de ces problèmes, nous donnons à présent les principaux résultats sur le sujet (nous les détaillerons ultérieurement). Pour MCDA, le ratio d'intégrité est $O(\sqrt{n})$ dans les graphes non orientés et $O(\sqrt{m})$ dans les graphes orientés, et il existe des familles de graphes planaires où tous les terminaux sont sur la face extérieure et où ce ratio est $\Omega(\sqrt{n})$. Pour MCM, le ratio d'intégrité est $O(\log|\mathcal{L}|)$ dans les graphes non orientés et $O(1)$ dans les graphes planaires, et il existe des familles d'instances (non planaires) où ce ratio est $\Omega(\log|\mathcal{L}|)$. Dans le cas orienté, le ratio est $O(\sqrt{n})$.

Enfin, nous verrons que, dans certains cas (c'est-à-dire, dans certains types de graphes), les matrices de contraintes de MFEM et MCM sont TU ; en conséquence, les problèmes sont polynomiaux dans ces cas-là.

Nous détaillons à présent quelques résultats basés sur (ou améliorant) la résolution par PL de MFEM et MCM.

La première approche que nous détaillons est celle de l'arrondi aléatoire, qui a été utilisée avec succès dans la conception d'algorithmes approchés pour un certain nombre de problèmes d'optimisation combinatoire (voir [93, Chapitre *Randomized approximation algorithms in combinatorial optimization* par R. Motwani, J.S. Naor et P. Raghavan (pp. 447 - 481)]). Elle consiste simplement à résoudre la relaxation continue du PLNE modélisant le problème, et à arrondir ensuite la solution optimale fractionnaire calculée pour

obtenir une solution entière. Néanmoins, cette phase d'arrondi est effectuée par le biais d'une méthode aléatoire : les parties fractionnaires des valeurs des variables dans la solution optimale calculée sont interprétées comme des probabilités pour arrondir à l'entier inférieur ou supérieur. L'objectif est de faire en sorte que, de cette façon, l'espérance de la valeur de la solution entière ainsi obtenue ne soit pas "trop éloignée" de la valeur de la solution fractionnaire initiale. Nous verrons cependant que ce genre d'approche est difficile à utiliser avec des problèmes comme MFEM, et ne peut être utilisée que sous certaines conditions : en effet, la phase d'arrondi peut mener à une solution non admissible. Enfin, il convient de noter que la phase d'arrondi aléatoire peut, parfois, être déterminisée, à l'aide de techniques sophistiquées (et souvent coûteuses en temps) [155]. En outre, comme nous l'avons déjà évoqué, les ratios d'approximation des algorithmes utilisant cette technique ne peuvent être meilleurs que le ratio d'intégrité.

Un certain nombre de résultats antérieurs portent également sur des études polyédrales de ces problèmes : en substance, elles consistent à rechercher des **inégalités valides**, c'est-à-dire des inégalités vérifiées par *toutes* les solutions entières mais pas (nécessairement) par toutes les solutions fractionnaires. À notre connaissance, dans les graphes non orientés, le polyèdre de MCM (c'est-à-dire, le polyèdre de sa formulation en PL) a surtout été étudié via celui de CMTM : Chopra et Rao [44], ainsi que Cunningham [53], donnent plusieurs familles d'inégalités valides pour ce problème. Bertsimas et al. ont donné des formulations non linéaires pour CMTM et MCM, et en ont déduit (par linéarisation) des formulations en PL pour ces deux problèmes, ainsi que quelques inégalités valides les concernant [22]. En outre, Chopra et Rao ont montré qu'une des familles d'inégalités valides qu'ils proposaient était suffisante pour rendre la formulation en PL de CMTM entière dans les arbres (c'est-à-dire que tout point extrême du polyèdre est entier). Ces inégalités sont appelées *inégalités d'arbres*, et sont définies ainsi [44] :

Définition 1.5. *Soit T un arbre non orienté et soit $\{t_1, \ldots, t_h\}$ un ensemble de terminaux. Supposons que les feuilles de T coïncident avec ces h terminaux. Alors, l'inégalité*

$$\sum_{a \text{ est une arête de } T} z_a \geq h - 1$$

est appelée **inégalité d'arbre** *pour T.*

Il n'est pas difficile de voir que ces inégalités sont effectivement valides pour CMTM. De surcroît, Chopra et Rao ont conçu dans [44] un algorithme efficace de type *Branch and Cut* (voir la section 1.1) pour résoudre CMTM, et qui est basé *uniquement* sur cette famille d'inégalités.

Pour MFEM, il n'existe à notre connaissance qu'une seule étude polyédrale : en 2000, Brunetta et al. [25] ont donné une famille très générale d'inégalités valides (qu'ils ont appelées *multi-handle comb inequalities* en anglais),

et les ont utilisées pour concevoir un algorithme de type *Branch and Cut* pour résoudre MFEM. Néanmoins, on peut remarquer que, pour FMTEM, il existe une famille très simple d'inégalités valides, que nous appellerons *inégalités d'ensembles impairs*. En voici une définition formelle :

Définition 1.6. *Soit $G = (S, A)$ un graphe non orienté et soit $\mathcal{T}$ un ensemble de terminaux. Pour chaque $X \subseteq S \setminus \mathcal{T}$, soit $(X, S \setminus X)$ l'ensemble des arêtes reliant X à $S \setminus X$ et soit $c(X, S \setminus X)$ la capacité totale de ces arêtes. On dit qu'une chaîne p_i* traverse *$(X, S \setminus X)$ si elle contient au moins une arête de $(X, S \setminus X)$. Alors, l'inégalité*

$$\sum_{i \text{ t. q. } p_i \text{ traverse } (X, S \setminus X)} f_i \leq \left\lfloor \frac{c(X, S \setminus X)}{2} \right\rfloor \tag{1.4}$$

est appelée **inégalité d'ensemble impair** *pour X.*

La validité de ces inégalités vient, en substance, du fait que, dans G, chaque unité de flot est routée à travers un nombre pair (qui peut être 0) d'arêtes de $(X, S \setminus X)$, puisque X ne contient aucun terminal. Donc, chaque unité de flot routée à travers une arête de $(X, S \setminus X)$ est comptée au moins deux fois, et la quantité totale d'unités de flot est donc égale à (au moins) $2 \sum_{i \text{ t. q. } p_i \text{ traverse } (X, S \setminus X)} f_i$. Cette quantité étant au plus $c(X, S \setminus X)$, on peut diviser les deux membres de l'inégalité par 2 et utiliser le caractère entier des f_i pour obtenir le résultat désiré. Au-delà de l'intérêt intrinsèque de ces inégalités, il existe un lien notable avec les inégalités d'arbres pour CMTM. Nous montrons à présent ce lien, et ignorons s'il avait déjà été fait auparavant. Considérons la famille d'inégalités suivantes :

Définition 1.7. *Soit T un arbre non orienté et soit $\mathcal{T} = \{t_1, \ldots, t_h\}$ un ensemble de terminaux. Supposons que les feuilles de T coïncident avec ces h terminaux. Alors, l'inégalité*

$$\sum_{i \text{ t. q. } p_i \text{ relie deux terminaux dans } \mathcal{T}} f_i \leq \left\lfloor \frac{\sum_{j \in \{1, \ldots, h\}} w(j)}{2} \right\rfloor$$

est appelée **inégalité d'arbre** *pour T, où, pour chaque j, $w(j)$ est le minimum des capacités des arêtes de la chaîne q_j reliant t_j à r_j, le sommet de degré au moins 3 dans T qui est le plus proche de t_j (voir la figure* 1.5*).*

Ces inégalités sont l'exacte correspondance, pour FMTEM, des inégalités d'arbres pour CMTM. On peut montrer qu'elles sont valides pour FMTEM en utilisant la validité des inégalités d'ensembles impairs :

Théorème 1.1. *Pour FMTEM, les inégalités d'arbres sont un cas particulier des inégalités d'ensembles impairs.*

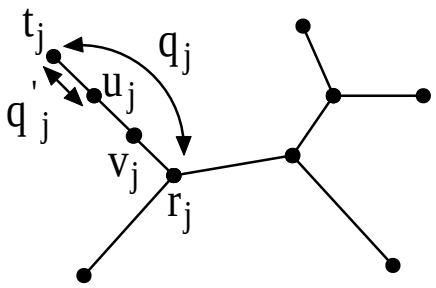


FIG. 1.5 – Une instance d'arbre ((u_j, v_j) a un poids $w(j)$).

Démonstration. On utilise les notations définies précédemment. Étant donné un arbre non orienté T, pour chaque j, on définit (u_j, v_j) comme étant une arête de poids $w(j)$ (u_j se trouvant sur la chaîne de t_j à v_j), et q'_j comme étant la chaîne de t_j à u_j (voir la figure 1.5). Alors, l'inégalité d'arbre pour T est obtenue en considérant l'inégalité d'ensemble impair associée à l'ensemble $X = S \setminus \bigcup_j q'_j$, puisque toute unité de flot doit être routée à travers au moins une arête (et, en réalité, à travers exactement deux arêtes) de $(X, S \setminus X)$, c'est-à-dire à travers (u_f, v_f) et (u_g, v_g) pour deux entiers f et g. □

Enfin, on peut citer deux travaux concernant la résolution pratique de MCM par des méthodes arborescentes.

D'abord, Létocart et Roupin [119] ont proposé une méthode arborescente basée sur la SDP pour résoudre des instances de MCM dans des arbres non orientés. En substance, la SDP est une modélisation plus générale que la PL, dont nous ne détaillerons pas les propriétés. Cependant, comme la résolution d'un SDP est souvent plus longue que celle d'un PL, il est important de noter que Roupin a proposé dans [147] une méthode générale pour construire une relaxation SDP à partir d'une relaxation PL, de façon à garantir que la borne de la relaxation SDP soit au moins aussi bonne que celle de la relaxation PL (ce qui montre que, en théorie, la SDP peut fournir de meilleures bornes).

Ensuite, Derhy a comparé, à l'aide d'un solveur commercial (CPLEX), plusieurs modélisations PL pour résoudre MCM dans le cas non orienté [55], et a observé des sauts d'intégrité assez faibles sur un ensemble d'instances générées aléatoirement à l'aide du logiciel *Rudy* [144] (alors que, en théorie, il existe des familles d'instances où ce saut est élevé).

Problèmes de décision

Historiquement, les premiers résultats sur les problèmes de chemins ou de flots avec plusieurs couples source-puits ont porté sur les versions de décision de ces problèmes (MFED, CDA et CDS). Tous sont $\mathcal{NP}$-complets, même dans des cas très particuliers (sauf, bien sûr, MFCD, qui est modélisable par

un PL en variables continues, et est donc polynomial). En effet, ils restent $\mathcal{NP}$-complets même lorsque $G+H$ est planaire et que le degré maximum est 3, ou lorsque $G+H$ est eulérien [128]. Ils sont également $\mathcal{NP}$-complets dans les grilles [114], et, pour CDA et MFED, dans les graphes séries-parallèles [132] (ce sont les graphes de largeur d'arbre 2). MFED reste $\mathcal{NP}$-complet pour $|\mathcal{L}| = 2$, même si les deux demandes sont bornées par le nombre de sommets [64] (en fait, même si l'une des deux vaut 1). Enfin, concernant les graphes orientés, CDA est $\mathcal{NP}$-complet même si $|\mathcal{L}| = 2$ [67].

Il existe cependant des cas polynomiaux particulièrement intéressants. Par exemple, lorsque $G + H$ est planaire et eulérien, MFED est polynomial [153]. Lorsque $G + H$ est planaire, Korach et Penn [110] et Frank et Szigeti [71] ont donné des conditions suffisantes de solubilité pour CDA (Srivastav et Stangier ont montré que, lorsque G est planaire mais pas $G + H$, ces conditions, même renforcées, ne garantissent pas la solubilité [156]) ; si, de plus, les terminaux sont situés sur un nombre borné de faces de G, CDA et CDS sont polynomiaux dans ce cas [128]. De même, lorsque $G + H$ est planaire et que $|\mathcal{L}|$ est fixé, MFED est polynomial [152]. En outre, Robertson et Seymour ont montré un résultat étonnant : lorsque $|\mathcal{L}|$ est fixé, CDA et CDS sont polynomiaux dans les graphes non orientés [146] (ce qui contraste avec le cas orienté). Enfin, étendant un résultat d'Okamura et Seymour [134] (voir ci-dessous), Frank a montré dans [70] (voir aussi [162]) que MFED était polynomial dans les graphes planaires, si tous les terminaux sont sur la face extérieure et si tous les sommets ne se trouvant pas sur la face extérieure ont un degré pondéré pair (ceci inclut les graphes planaires superficiels, qui forment une sous-classe des graphes séries-parallèles). Dans les graphes orientés, CDA est polynomial (mais pas FPT [154]) si $|\mathcal{L}|$ est fixé et si le graphe est un GSC [67], ou si $|\mathcal{L}| = 2$ et si le graphe support est eulérien [73].

Il faut noter qu'un certain nombre de cas polynomiaux (et notamment le théorème d'Okamura et Seymour) sont liés à la suffisance (dans ces cas-là) d'une condition nécessaire naturelle, appelée **condition de coupe**. Nous la détaillerons en section 5.2, mais elle peut s'énoncer brièvement ainsi :

$$\forall X \subseteq S, |\delta_G(X)| \geq |\delta_H(X)|, \text{ où } \delta_H(X) = \{(s_i, s'_i) \in \mathcal{L} \ / \ |\{s_i, s'_i\} \cap X| = 1\}$$

Il est aisé de montrer que c'est une condition nécessaire (mais montrer sa suffisance dans certains cas l'est beaucoup moins) :

Théorème 1.2 (voir [134]). *La condition de coupe est nécessaire à l'existence d'une solution pour CDA.*

Démonstration. Supposons qu'il existe une solution, et un ensemble $X \subseteq S$ tel que $|\delta_G(X)| < |\delta_H(X)|$. Alors, dans la solution, $|\delta_H(X)|$ chemins "sortent" de X (d'après la définition de $\delta_H(X)$), mais seules $|\delta_G(X)|$ ($< |\delta_H(X)|$) arêtes sont disponibles : par conséquent, au moins deux chemins empruntent la même arête, ce qui contredit le fait que c'est une solution pour CDA. □

Nous ne pouvons pas détailler tous les résultats liés à l'ensemble de ces problèmes : ils sont innombrables. Pour de plus amples informations, le lecteur pourra se référer à [151, Partie VII] et [112, 160, 161]. Nous détaillerons le théorème d'Okamura et Seymour, ainsi que d'autres résultats liés à ces problèmes de décision dans des grilles, en section 5.2.

Complexité et approximation des problèmes de chemins

Les problèmes d'optimisation (MFEM/MFEMI, MCDA/MLCDA, MCDS et FMTEM) associés aux problèmes de décision précédents étant au moins aussi difficiles que ces derniers, il faut se placer sous des hypothèses encore plus restrictives pour espérer trouver des cas polynomiaux.

On se place d'abord du point de vue de la complexité. MFEM et MFEMI sont $\mathcal{NP}$-difficiles (et même APX-difficiles) dans les arbres (même lorsque toutes les arêtes sont pondérées par 1 ou 2 [80] ou que toutes les capacités sont $\Omega(\log m)$ [156]), mais polynomiaux (la matrice des contraintes de la formulation en PL étant TU [52]) dans les chaînes et les **arbres orientés** [50], c'est-à-dire dans les arbres où l'on peut orienter chaque arête (une orientation par arête) de façon à ce qu'il existe un chemin de s_i à s'_i, pour tout i. Ils sont également polynomiaux dans les étoiles [80].

En outre, pour $|\mathcal{L}| = 2$, MCDA (et donc MFEM) est $\mathcal{NP}$-difficile dans les graphes non orientés et dans les GSC [64] (MLCDA, et donc MFEMI, étant $\mathcal{NP}$-difficile dans les graphes orientés [67]), alors qu'il existe un algorithme presque linéaire si $G + H$ est planaire et non orienté [111]. En fait, dans les graphes non orientés, MCDA est même APX-difficile si $|\mathcal{L}| = 2$ [89]. À notre connaissance, le statut de MCDA dans les graphes planaires est par contre ouvert si $|\mathcal{L}|$ est fixé (alors que MCDS est polynomial dans ce cas [82]). Néanmoins, on peut remarquer que, dans les graphes non orientés, lorsque $|\mathcal{L}|$ est fixé, MLCDA est polynomial (MCDA l'est également si le graphe est de degré maximum borné) : il suffit d'appeler un nombre fixé de fois l'algorithme de Robertson et Seymour pour résoudre CDA [146].

MCDA et MLCDA sont polynomiaux dans les anneaux [123], et, contrairement à MFEM, dans les arbres [80], mais la preuve de APX-difficulté pour MFEM dans les arbres avec des poids 1 et 2 montre qu'ils deviennent $\mathcal{NP}$-difficiles (et même APX-difficiles) dans les cactus : il suffit pour cela de remplacer, dans la preuve, chaque arête de poids 2 par deux chaînes parallèles de longueur 2 (ne contenant que des arêtes de poids 1). Cette preuve permet même de montrer que MCDA est APX-difficile dans les arbres d'anneaux, en remplaçant toute arête (u, v) de poids 1 qui n'est pas dans un cycle par un cycle (u, v, w, u) (où w est un nouveau sommet) contenant trois arêtes de poids 1, puis en ajoutant deux liaisons (u, w) et (v, w). Erlebach montre que MLCDA est aussi APX-difficile dans les arbres d'anneaux [61], et dans les arbres bi-orientés [62] (où il devient polynomial si le degré maximum est borné). En outre, il montre que MLCDA est $\mathcal{NP}$-difficile dans les graphes

complets [63] : la preuve part d'une instance de MLCDA dans un graphe quelconque, et ajoute les arêtes (u, v) nécessaires pour rendre le graphe complet, ainsi que toutes les *liaisons* (u, v) associées. Il est alors facile de montrer que l'instance obtenue est équivalente à l'instance initiale.

Il convient de remarquer que MFEM reste $\mathcal{NP}$-difficile dans les graphes planaires superficiels qui sont 2-arête-connexes et qui ont tous leurs degrés pairs. En effet, en partant de la $\mathcal{NP}$-difficulté de MCDA dans les arbres d'anneaux (des graphes planaires superficiels dont les degrés sont pairs) détaillée précédemment, on peut remplacer tout sommet qui n'est pas dans un cycle par un cycle (ayant un nombre pair de sommets) dont les arêtes sont pondérées alternativement par un grand entier $N \geq 3n$ et par $N + 1$, et qui a autant de sommets que le sommet initial avait d'arêtes adjacentes. La i^e arête initialement adjacente au sommet est alors reliée au i^e sommet du nouveau cycle (voir aussi la section 6.3 du chapitre 6). On obtient ainsi un graphe planaire superficiel qui est 2-sommet-connexe et dont les degrés (pondérés) sont pairs (le degré pondéré des sommets des nouveaux cycles valant $1 + N + (N + 1) = 2(N + 1)$). Cependant, cette transformation nécessite d'utiliser de grands poids et ne peut donc être appliquée à MCDA : ce cas particulier semble donc ouvert pour MCDA.

On peut également mentionner une série d'articles concernant des propriétés des chemins disjoints dans les graphes de forte expansion (*expanders* en anglais). Cependant, nous ne détaillerons pas ces résultats, qui sont très spécifiques à la structure de ces graphes, et ne nous seront d'aucune utilité. Le lecteur intéressé pourra se référer à [75, 137].

Enfin, concernant FMTEM dans les graphes non orientés, le problème est connu pour être polynomial dans le cas non pondéré, où il peut se résoudre à l'aide d'un algorithme combinatoire issu de la théorie des *matroïdes* [151, p. 1283] (et qui fournit également un algorithme pseudo-polynomial pour le cas général). Des algorithmes combinatoires polynomiaux ont également été proposés pour résoudre des cas particuliers : les arbres [51] et les graphes où tous les sommets non terminaux ont un degré pondéré pair [72]. Dans le cas orienté, un seul résultat de complexité semble connu : Costa et al. [52] montrent que FMTEM est polynomial dans les GSC (en se ramenant à un simple problème de flot maximum). Cependant, nous verrons au chapitre 8 que des résultats de complexité pour FMTEM peuvent être déduits assez naturellement des résultats existants pour MFEM.

Nous examinons à présent les résultats connus concernant l'approximation de l'ensemble de ces problèmes. On peut noter que, en règle générale, ils se comportent "assez mal" de ce point de vue.

Une série d'articles a été est consacrée à l'étude de l'approximation de ces problèmes dans les graphes généraux. Avant 2003, deux algorithmes $O(\sqrt{m})$-approchés étaient connus pour MCDA (nous décrirons les résultats ultérieurs en section 1.4) : un algorithme glouton (noté SPF, pour *Shortest Paths First* [103, 107]), que nous décrivons plus bas (pour sa grande simplicité et l'analyse

typique de sa performance), et un algorithme basé sur des méthodes d'arrondi aléatoire [13] (des variantes sont disponibles pour MFEM et MFEMI). De façon étonnante, le meilleur résultat d'inapproximabilité connu pendant longtemps pour les graphes non orientés était que MCDA était APX-difficile [80]. Dans les graphes orientés, un résultat beaucoup plus fort est connu depuis 1999 : MCDA est $\mathcal{NP}$-difficile à approcher à un ratio $m^{1/2-\epsilon}$, pour tout $\epsilon > 0$ [89]. Donc, du point de vue de l'approximation, le statut de MCDA dans les graphes orientés semble réglé (voir la section 1.4). En outre, il est important de noter qu'il existe des familles de graphes planaires (non orientés) où le ratio d'intégrité est $\Theta(\sqrt{m})$ [80].

Pour des classes de graphes plus restreintes, il existe des algorithmes approchés ayant des ratios d'approximation constants : pour MFEM, un algorithme 2-approché dans les arbres [80] (basé sur une approche primale-duale, et que nous détaillons en section 2.1.1) ; pour MLCDA, un algorithme 9-approché dans les graphes complets [28] et un algorithme 3-approché dans les arbres d'anneaux [61] ; pour MCDA, un algorithme $\left(\frac{5}{3}+\epsilon\right)$-approché dans les arbres bi-orientés [62], un algorithme $O(1)$-approché dans une classe de graphes généralisant les grilles rectangulaires [102, 104] et un algorithme $O(f^*)$-approché dans les graphes où un certain paramètre $\mathcal{F}$ (appelé *flow number* en anglais ; voir [108] pour les détails) vaut f^*. De surcroît, pour les réseaux de grandes capacités (c'est-à-dire, les graphes où toutes les capacités sont $\Omega(\log m)$), un algorithme $O(1)$-approché peut être obtenu pour MFEM en appliquant des techniques d'arrondi aléatoire [140].

Enfin, Garg et al. donnent un algorithme $2\log_2(|\mathcal{L}|)$-approché pour FMTEM dans les graphes orientés (c'est en fait une conséquence de leur algorithme dans [78], qui renvoie un flot multiterminal entier).

Pour de plus amples informations, nous renvoyons le lecteur à une série d'états de l'art sur le sujet [5, 52, 77, 103].

L'algorithme SPF. Pour clôturer cette section sur les problèmes de chemins, nous détaillons brièvement le fonctionnement de l'algorithme glouton SPF (que nous utiliserons au chapitre 7) et l'analyse (très simple) de sa performance. On peut résumer le déroulement de cet algorithme ainsi. SPF :

1. Choisir, puis retirer de P, le plus court chemin (au sens du nombre d'arêtes) $p^* \in P$; Si $P = \emptyset$, stopper et renvoyer les chemins choisis ;
2. Retirer du graphe toutes les arêtes de p^*, et retirer de P tout chemin qui traverse une de ces arêtes ; Revenir en 1.

Une façon d'implémenter cet algorithme pour le rendre polynomial est de calculer un plus court chemin de s_i à s'_i pour tout i.

Théorème 1.3 ([103, 107])**.** *SPF est un algorithme $\lceil\sqrt{m}\,\rceil$-approché pour le problème MCDA.*

Démonstration. La preuve repose sur une idée très simple : il suffit de montrer que, au moment où SPF choisit son i^e chemin (appelons-le p_i^*), il aurait pu choisir, à la place, au plus $\lceil \sqrt{m} \rceil$ chemins de la (ou d'une) solution optimale. En effet, si p_i^* est de longueur au plus $\lceil \sqrt{m} \rceil$, alors il a une intersection avec au plus $\lceil \sqrt{m} \rceil$ chemins, et donc avec au plus $\lceil \sqrt{m} \rceil$ chemins d'une solution optimale. Sinon, c'est qu'il ne reste plus aucun chemin de longueur au plus $\lceil \sqrt{m} \rceil$ dans P (puisqu'on choisit le plus court), et donc il ne peut rester plus de $\lceil \sqrt{m} \rceil$ chemins dans la solution optimale (car $\lceil \sqrt{m} \rceil \cdot \lceil \sqrt{m} \rceil \geq m$). □

Complexité et approximation des problèmes de coupe

MCM est polynomial dans les chaînes et les arbres orientés [52], mais devient $\mathcal{NP}$-difficile (et même APX-difficile) dans les étoiles non pondérées (néanmoins, il peut être résolu en temps polynomial dans les arbres si $|\mathcal{L}|$ est fixé) [80], alors que MFEM y est polynomial. Nous détaillons la preuve de complexité pour MCM plus bas, car elle utilise plusieurs idées qui nous seront utiles ultérieurement, et en particulier dans le chapitre 5. En outre, pour $|\mathcal{L}| = 2$, MCM est polynomial dans les graphes non orientés (par deux applications d'un algorithme de coupe minimum [163]) et APX-difficile dans les graphes orientés (car CMTM l'est [81]). Pour $|\mathcal{L}| = 3$, CMTM et MCM deviennent $\mathcal{NP}$-difficiles et même APX-difficiles dans les graphes non orientés [54]. CMTM est $\mathcal{NP}$-difficile dans les graphes planaires, où il devient polynomial si $|\mathcal{L}|$ est fixé [54, 91, 164]. De surcroît, d'après [54], CMTM est polynomial dans les graphes de largeur d'arbre bornée (en utilisant des techniques standard de programmation dynamique ; voir [14]), et, d'après [42], il est également polynomial dans les graphes planaires si tous les terminaux sont sur la face extérieure (le cas des arbres ayant déjà été traité dans [44, 51, 58, 59, 60] et le cas des cactus triangulaires dans [22]) : les auteurs montrent que, dans ce cas-là, CMTM est *équivalent* au problème de l'*arbre Steiner de poids minimum* ([76]) avec tous les terminaux sur la face extérieure. Citons, enfin, le fait que Provan et Burk ont montré que MCM est polynomial si $G + H$ est planaire et que tous les terminaux sont sur la face extérieure de G [138].

Il faut également noter que MCM et ses variantes se comportent mieux, du point de vue de l'approximation, que les problèmes de chemin associés ; néanmoins, pour les cas les plus généraux, les ratios restent appréciables.

Concernant MCM, il existe un algorithme 2-approché dans les arbres [80], un algorithme $O(1)$-approché dans les graphes planaires non orientés [157] et un algorithme $O(\log |\mathcal{L}|)$-approché dans les graphes non orientés [79] (dans les trois cas, le ratio d'intégrité est en outre borné par le ratio d'approximation). On peut noter qu'il existe également un algorithme $2\alpha(H)\left(1 - \frac{1}{|\mathcal{T}|}\right)$-approché pour MCM dans les graphes non orientés (où $\alpha(H)$ est le nombre de stabilité de H et $\mathcal{T}$ est l'ensemble des terminaux, voir plus bas) [22]. Par

ailleurs, MCM admet un PTAS dans les graphes non orientés et non pondérés de largeur d'arbre et de degré maximum bornés (où il reste $\mathcal{NP}$-difficile), alors que la suppression d'une seule de ces trois hypothèses (non pondéré, degré maximum borné, largeur d'arbre bornée) rend le problème APX-difficile [27]. Dans les graphes orientés, il existe un algorithme $O(\sqrt{n})$-approché pour MCM [88] (voir aussi [43, 113]), ainsi qu'un algorithme $O(|\mathcal{L}|)$-approché trivial (calculer, pour chaque i, une coupe minimum entre s_i et s'_i, puis prendre l'union de toutes ces coupes).

Concernant CMTM, le premier algorithme approché (nommé CIO pour *algorithme par Confinements Individuels Optimaux*) fut conçu par Dahlhaus et al. [54] : il s'agit d'un algorithme $2\left(1-\frac{1}{|\mathcal{T}|}\right)$-approché que nous détaillons plus bas, car il est très simple à énoncer et à analyser, et a un lien notable avec l'algorithme donné dans [22] pour MCM. Il faut noter que Garg et al. ont ensuite donné un algorithme avec le même ratio d'approximation pour CMTMS, via une approche totalement différente (celle utilisée pour CMTM ne pouvant être adaptée à CMTMS) [78, 81]. Un algorithme avec un ratio d'approximation légèrement inférieur à 1.3438 et faisant appel à des techniques beaucoup plus sophistiquées a ensuite été trouvé pour CMTM [26, 97]. Dans les graphes orientés, il existe un algorithme 2-approché pour CMTM [130] (voir aussi [78]). Enfin, les auteurs de [54] ont remarqué que, dans les graphes non orientés, les algorithmes approchés pour CMTM peuvent être convertis en algorithmes approchés pour MCM (avec le même ratio) lorsque le nombre de liaisons est *fixé* (il suffit d'énumérer l'ensemble des partitionnements possibles des terminaux ; cette méthode est détaillée en section 3.1).

Le lecteur intéressé par plus de détails concernant ces problèmes pourra se reporter aux références suivantes : [5, 52, 77] et [93, Chapitre *Cut problems and their application to divide-and-conquer* par D.B. Shmoys (pp. 192–235)].

Preuve de complexité pour MCM dans les étoiles [80]. Cette preuve est particulièrement instructive, car elle constitue un exemple typique de réduction pour MCM. Comme nous le ferons pour d'autres cas particuliers de MCM, nous allons montrer que le problème CouvertureParLesSommets se réduit polynomialement à MCM dans les étoiles non pondérées. Pour cela, considérons une instance $\mathcal{I}$ de CouvertureParLesSommets [76] sur un graphe $G_{\mathcal{I}} = (S_{\mathcal{I}}, A_{\mathcal{I}})$ à n sommets : étant donné un entier N, le problème consiste à décider s'il existe pour $G_{\mathcal{I}}$ une **couverture** de taille au plus N, où une couverture est un ensemble $C_{\mathcal{I}} \subseteq S_{\mathcal{I}}$ tel que, pour tout $(u, v) \in A_{\mathcal{I}}$, on a $u \in C_{\mathcal{I}}$ ou $v \in C_{\mathcal{I}}$. On va construire une instance de MCM avec un graphe support G et un graphe de demandes H. G est une étoile à n feuilles (et donc $n+1$ sommets en tout), et les terminaux (les sommets de H) sont les feuilles. En outre, H est égal à $G_{\mathcal{I}}$. On définit l'équivalence suivante entre les solutions des deux instances : une arête de G est coupée ssi elle est adjacente à une feuille correspondant à un sommet de la couverture. Alors, il est facile

de voir qu'il existe une multicoupe de taille au plus N pour l'instance de MCM ssi il existe une couverture de taille au plus N dans $\mathcal{I}$. On a donc :

Théorème 1.4 ([80]). *MCM est APX-difficile dans les étoiles non pondérées.*

Notons que la réduction utilisée préserve le ratio d'approximation : cela implique qu'un algorithme α-approché pour MCM dans ce cas fournirait un algorithme α-approché pour CouvertureParLesSommets. Comme la question de trouver un algorithme ayant un ratio d'approximation meilleur que 2 pour ce problème est ouverte depuis longtemps, il semble difficile d'améliorer l'algorithme 2-approché pour MCM dans les arbres [80].

L'algorithme *CIO* pour CMTM. Pour finir, nous détaillons brièvement l'algorithme *CIO* et l'analyse (très simple) de sa performance.

Nous donnons d'abord le déroulement de cet algorithme. Rappelons qu'il prend en entrée un graphe G non orienté et un ensemble $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$ de sommets terminaux. L'algorithme *CIO* peut alors se décrire ainsi :

1. Calculer une coupe minimum C_i séparant t_i de $\mathcal{T} \setminus \{t_i\}$, pour tout i ;
2. Renvoyer la coupe multiterminale $C = \bigcup_{i \neq j} C_i$, où j est tel que $\|C_j\| = \max_i \|C_i\|$.

Théorème 1.5 ([54]). *Dans les graphes non orientés, l'algorithme CIO est un algorithme* $2\left(1 - \frac{1}{|\mathcal{T}|}\right)$*-approché pour CMTM.*

Démonstration. Soit C^* une solution optimale pour CMTM, et soit $C_i^* \subseteq C^*$ l'ensemble des arêtes qui séparent t_i de $\mathcal{T} \setminus \{t_i\}$ dans C^*. On a $\|C_i\| \leq \|C_i^*\|$ pour tout i, car C_i est une coupe minimum. En outre, $\|C\| \leq \sum_i \|C_i\| - \max_i \|C_i\| \leq \sum_i \|C_i\| - \frac{1}{|\mathcal{T}|} \sum_i \|C_i\|$. Enfin, pour toute arête $a = (u, v)$ dans C^*, on a $u \in C_f^*$ et $v \in C_g^*$ pour $f \neq g$, et donc $\sum_i \|C_i^*\| = 2\|C^*\|$ (chaque arête est comptée deux fois). En conclusion, on a donc : $\|C\| \leq \sum_i \|C_i\| \left(1 - \frac{1}{|\mathcal{T}|}\right) \leq \sum_i \|C_i^*\| \left(1 - \frac{1}{|\mathcal{T}|}\right) = 2\left(1 - \frac{1}{|\mathcal{T}|}\right) \|C^*\|$, ce qui termine la preuve. □

Il convient de remarquer que le résultat donné dans [22] généralise en fait ce ratio d'approximation. Pour le montrer, commençons par définir le nombre de stabilité d'un graphe : c'est le cardinal du plus grand **stable** du graphe. Étant donné un graphe $G = (S, A)$, un stable (ou ensemble indépendant) de G est un ensemble $J \subseteq S$ tel que, pour toute arête $a \in A$, *au plus* une des deux extrémités de a est dans J.

Comme nous l'avons mentionné plus haut, les auteurs de [22] donnent un algorithme $2\alpha(H)\left(1 - \frac{1}{|\mathcal{T}|}\right)$-approché pour MCM dans les graphes non orientés, où $\alpha(H)$ est le nombre de stabilité de H et $\mathcal{T}$ est l'ensemble des

terminaux. Or, on peut remarquer que, lorsque H est complet (ce qui correspond au cas d'une instance de MCM avec un graphe de demandes complet, c'est-à-dire à une instance de CMTM), on a $\alpha(H) = 1$, et on retrouve donc le ratio d'approximation de l'algorithme CIO.

1.4 Résultats de cette thèse ou concomitants à cette thèse

Dans cette section, nous décrivons l'ensemble des résultats de cette thèse. Auparavant, nous détaillons tous les résultats publiés sur le sujet depuis le début de la thèse (octobre 2003), pour donner un aperçu clair des évolutions de l'état de l'art au cours de ces dernières années.

1.4.1 Résultats récents

Détaillons d'abord les résultats en rapport avec les problèmes de chemins.

Concernant l'approximation de MCDA dans les graphes généraux ou planaires, des avancées notables ont été réalisées : Chekuri et Khanna d'abord [32], puis Varadarajan et Venkataraman [158], ont considérablement amélioré l'analyse du ratio d'approximation de SPF dans les graphes orientés ou non (voir aussi [90]) ; au final, un algorithme $O(n^{2/3}\log^{2/3} n)$-approché pour MCDA dans les graphes orientés est obtenu. Il faut noter que, pour $m = \Omega(n^{\frac{4}{3}+\epsilon})$, c'est un ratio bien meilleur que le précédent ratio $O(\sqrt{m})$, pour tout $\epsilon > 0$. En outre, cela montre que l'approximabilité de MCDA dans les graphes orientés n'est pas encore complètement réglée, puisque le résultat d'inapproximabilité de Guruswami et al. [89], qui stipule que MCDA est $\mathcal{NP}$-difficile à approcher à un ratio $m^{\frac{1}{2}-\epsilon}$ pour tout $\epsilon > 0$, a été montré dans une famille d'instances où $m = \Theta(n)$. Donc, *comme fonction de* n, il reste encore, dans les graphes orientés, à réduire l'écart entre la borne supérieure ($O(n^{2/3}\log^{2/3} n)$) et la borne inférieure ($\Omega(n^{\frac{1}{2}-\epsilon})$). Très récemment, Nguyen [131] et, indépendamment, Chekuri et al. [39], ont même montré qu'il existait un algorithme $O(\sqrt{n})$-approché pour MCDA dans les graphes non orientés et les GSC. Cependant, rappelons que le meilleur résultat d'inapproximabilité connu pour MCDA dans les graphes non orientés a longtemps été son APX-difficulté [80]. Il aura fallu attendre 2005 pour un résultat beaucoup plus fort : Andrews et al. [7, 8] ont montré que, pour tout $\epsilon > 0$, il n'existe pas d'algorithme $O(\log^{\frac{1}{2}-\epsilon})$-approché pour MCDA dans les graphes non orientés si une hypothèse de complexité un peu plus forte que $\mathcal{P} \neq \mathcal{NP}$, que nous ne détaillerons pas davantage, est vraie. Cependant, à nouveau, il reste un écart important entre les bornes inférieure et supérieure. Pour y remédier, Chekuri, Khanna et Shepherd ont, dans une série d'articles, tenté de réduire le ratio d'approximation à un ratio logarithmique ou constant à l'aide d'un nombre d'hypothèses supplémentaires très restreint.

Ils ont d'abord obtenu le résultat remarquable qu'il existait un algorithme $O(\log n)$-approché pour MCDA dans les graphes planaires où la capacité minimum vaut 2 [35, 37] (en outre, Rao et Zhou ont conçu un algorithme $O(\log^{10} n)$-approché pour MCDA dans les graphes où toute coupe entre toute paire de sommets possède $\Omega(\log^5 n)$ arêtes [143], qui est fondé en partie sur ces travaux de Chekuri et al.). Ils ont ensuite amélioré ce résultat sous une hypothèse à peine plus forte : il existe un algorithme $O(1)$-approché pour MCDA dans les graphes planaires où la capacité minimum vaut 4 [40]. Il convient de remarquer que les deux résultats utilisent des techniques très différentes, que nous ne détaillerons pas. Néanmoins, dans [40], Chekuri et al. montrent un résultat intermédiaire intéressant : il existe un algorithme $O(1)$-approché pour MCDA dans les graphes étudiés dans [70], c'est-à-dire les graphes planaires où tous les terminaux sont sur la face extérieure et tous les sommets qui ne sont pas sur la face extérieure ont un degré pondéré pair. Leur approche, par arrondi d'une solution fractionnaire, montre également que le ratio d'intégrité est $O(1)$. Ceci répond à une question laissée ouverte dans une version préliminaire d'un article rédigé dans le cadre de la présente thèse [16]. En outre, cela implique qu'il existe un algorithme $O(1)$-approché pour MFEM dans les graphes planaires où tous les terminaux sont sur la face extérieure et où toutes les capacités sont au moins 2. En effet, étant donné un tel graphe G, on peut diminuer chaque capacité impaire de 1 et obtenir un graphe G' qui n'a que des capacités paires ; on peut ensuite appliquer l'algorithme de Chekuri et al. Il n'est pas difficile de voir que le ratio d'intégrité dans G est au plus $\frac{3}{2}$ fois le ratio d'intégrité dans G', puisque la valeur d'une multicoupe fractionnaire minimum (et donc d'un multiflot fractionnaire maximum) dans G est au plus $\frac{3}{2}$ fois la valeur dans G' (le pire cas se produisant lorsque toutes les capacités sont diminuées de 3 à 2). Enfin, on peut noter que Kleinberg [105] a montré qu'il existait un algorithme $O(\log^2 n)$-approché pour MCDA dans les graphes planaires eulériens (et même si, comme dans les grilles, tous les sommets d'une des faces n'ont pas un degré pair) ; son résultat utilise l'algorithme de Chekuri et al. [35, 37].

On peut également citer quelques résultats concernant des graphes où certains paramètres sont bornés. Par exemple, Obata a montré, à l'aide d'une variante de SPF, qu'il existait un algorithme $O(\log(|\mathcal{L}|))$-approché (resp. $O(1)$-approché) dans les graphes (resp. dans les graphes planaires) où toute multicoupe a une valeur $\Omega(\sum_{a\in A} c(a))$ [133]. Cependant, comme nous le verrons en section 2.1.2, cette condition peut ne pas être vérifiée même dans certaines familles d'arbres. Nous avons montré dans [15] que le ratio d'intégrité pour MCDA était $O(1)$ dans une sous-classe des graphes planaires à k niveaux (k borné), et que MCDA était polynomial dans les graphes de nombre cyclomatique borné. Nous avons également montré que le ratio d'intégrité pour MFEM était $O(1)$ dans les graphes de nombre cyclomatique

borné, et avons laissé comme ouvert le problème suivant : existe-t-il un algorithme $O(1)$-approché pour MFEM dans les graphes de largeur d'arbre bornée ? Plus récemment, il a été montré que, dans ces graphes, il existe un algorithme $O(\log n)$-approché pour MCDA [41].

Par ailleurs, Marx [125] a montré la $\mathcal{NP}$-complétude de CDA dans les grilles eulériennes (c'est-à-dire dans les grilles où $G+H$ est eulérien). Concernant FMTEM, Keijsper et al. [99] ont récemment montré qu'il était polynomial dans les graphes non orientés, à l'aide de la méthode des ellipsoïdes (le résultat est fondé sur le fameux théorème de Mader sur les $\mathcal{T}$-chemins [151, Chap. 73]) : en substance, ils montrent que les inégalités (1.4) sont suffisantes (avec les contraintes classiques) pour garantir l'existence d'une solution entière optimale.

Détaillons à présent les résultats concernant les problèmes de coupe.

Récemment, Chekuri et al. [36] ont étudié une relaxation bi-orientée de CMTM (c'est-à-dire, dans le graphe bi-orienté obtenu à partir du graphe non orienté initial en remplaçant chaque arête par deux arcs opposés), et l'ont comparée à la relaxation utilisée dans [26]. Guo et Niedermeier ont, quant à eux, montré que MCM était FPT dans les arbres si le paramètre est le nombre d'arêtes à enlever [86], alors que Marx a obtenu une série de résultats concernant le caractère FPT de plusieurs variantes de MCM et MCMS [126]. On peut aussi noter des résultats remarquables concernant la complexité et l'approximabilité de MCM. D'abord, Agarwal et al. ont montré qu'une relaxation SDP naturelle pour MCM avait un ratio d'intégrité $\Omega(\log n)$ dans les graphes non orientés, prouvant par là-même que cette formulation SDP ne pouvait être d'aucun secours pour améliorer le meilleur ratio d'approximation connu [2]. Saks et al. ont également montré que, dans les graphes orientés, la formulation en PL de MCM avait un ratio d'intégrité $\Omega(|\mathcal{L}|)$ [148], et ne pouvait donc pas servir, elle non plus, à concevoir de "bons" algorithmes approchés (il faut cependant remarquer que, dans la famille d'instances utilisée par les auteurs, on a $|\mathcal{L}| = \Theta(\log n)$). Ensuite, Chawla et al. ont montré qu'il n'existe pas d'algorithme $O(1)$-approché pour MCM si la *Unique Games Conjecture* est vraie [31]. C'est le premier résultat d'inapproximabilité pour MCM qui soit plus fort que l'APX-difficulté ; néanmoins, il repose sur une conjecture différente de $\mathcal{P} \neq \mathcal{NP}$. Enfin, il a été montré par les auteurs de [87], et, indépendamment, par nous dans le cadre de cette thèse [16], que MCM est polynomial dans les graphes de largeur d'arbre bornée lorsque le nombre de liaisons est fixé. On peut également citer le fait que de nouveaux algorithmes 2-approchés ont été obtenus pour MCM dans les arbres, basés sur une approche d'arrondi d'une solution fractionnaire [84, 121] (contrairement à l'algorithme de Garg et al. [80]).

1.4.2 Résultats et organisation de la thèse

Présentation des résultats

Une partie importante de notre travail a consisté à généraliser les résultats connus et à rechercher des paramètres intéressants qui, s'ils sont fixés, rendent les problèmes plus faciles à résoudre de façon exacte ou approchée.

Nos premiers résultats concernent les grilles, où nous résolvons deux cas particuliers polynomiaux, le premier à l'aide d'une approche primale-duale et d'un certain nombre de résultats antérieurs [69, 134], le second à l'aide d'un algorithme glouton. Nous donnons également des résultats de complexité concernant MCM et MFEMI, qui montrent que ces problèmes restent $\mathcal{NP}$-difficiles même dans des cas très particuliers de grilles. Ces résultats ont fait l'objet de deux publications en revue [18, 19].

Un deuxième ensemble de travaux concerne la généralisation de la totalité des résultats de Garg et al. dans les arbres [80]. En premier lieu, nous montrons que MCDA est polynomial dans les graphes de nombre cyclomatique borné, et que le ratio d'intégrité pour MFEM est borné dans ces graphes (nous donnons aussi un algorithme approché fondé sur une approche primale-duale). De surcroît, nous donnons un algorithme approché (également fondé sur une approche primale-duale) avec un ratio borné pour MCDA dans les graphes planaires à k niveaux d'arêtes (k borné). Ces graphes, qui généralisent les cactus, forment une sous-classe naturelle des graphes planaires à k niveaux et n'ont jamais été étudiés à notre connaissance. Une propriété importante est qu'ils sont de largeur d'arbre bornée mais quelconque (comme les graphes planaires à k niveaux), et qu'ils incluent les grilles planaires ayant une des deux dimensions bornée.

Concernant ces résultats, il faut remarquer que, d'une part, MCDA est APX-difficile dans les cactus triangulaires (et même dans les arbres d'anneaux), et il y a donc peu de chances pour qu'il existe une classe de graphes plus générale que la nôtre où MCDA serait polynomial, et que, d'autre part, les seules classes de graphes pour lesquelles il existait auparavant des algorithmes polynomiaux pour MCDA étaient les chaînes, les anneaux et les arbres (trois classes de graphes de nombre cyclomatique inférieur ou égal à 1). Par ailleurs, notre résultat pour MFEM est la première tentative de généralisation de l'algorithme approché à ratio constant pour MFEM dans les arbres, et, comme il utilise une approche primale-duale, il montre également que le ratio d'intégrité pour MFEM est borné dans ce cas (alors que, rappelons-le, le ratio d'intégrité pour MCDA peut être $\Omega(\sqrt{n})$, même dans les graphes planaires). Enfin, il faut noter que tous ces résultats ont été obtenus alors qu'il n'existait aucun résultat concernant MCDA dans des graphes de largeur d'arbre bornée mais quelconque (les résultats de [41] étant pos-

térieurs aux nôtres), et qu'ils ont fait l'objet d'une publication dans une conférence internationale [15].

Nous avons également quelques résultats concernant MFEM dans les anneaux. Ce problème est polynomial dans les chaînes mais $\mathcal{NP}$-difficile dans les arbres : nous prouvons qu'il est polynomial dans les anneaux et étudions quelques aspects du polyèdre de sa formulation en PL.

Nous montrons par ailleurs que, lorsque le nombre de liaisons est fixé, MCM est polynomial dans les graphes de largeur d'arbre bornée et dans les graphes planaires ayant leurs terminaux sur la face extérieure (le cas des arbres étant déjà connu [80]). Le premier résultat a été démontré indépendamment dans [87] : notre approche est plus simple et plus rapide, mais moins générale (car d'autres résultats annexes sont obtenus dans [87]). Nous prouvons également qu'un algorithme conçu par Yeh [164] pour résoudre CMTM dans les graphes planaires ayant un nombre de liaisons fixé est erroné. Nous avons été amené à examiner cet algorithme alors que nous essayions de l'adapter à un cas (à peine) plus général, et avons découvert une erreur dans sa preuve à cette occasion. Remarquons qu'il existe d'autres algorithmes (qui, jusqu'à preuve du contraire, sont justes) pour résoudre CMTM dans ce cas [54, 91], mais ils sont relativement compliqués (surtout comparés à celui de Yeh).

Nous fournissons ensuite toute une série de résultats concernant la complexité et l'approximabilité de MCM dans les graphes de largeur d'arbre bornée. En particulier, nous montrons que MCMS est $\mathcal{NP}$-difficile dans les cactus triangulaires non pondérés de degré maximum et de largeur de chaîne bornés, alors qu'il existe un PTAS dans ce cas et que le problème est polynomial dans les arbres non pondérés. Nous montrons également que MCM est $\mathcal{NP}$-difficile dans les GSC et APX-difficile dans les graphes orientés non pondérés de largeur d'arbre et de degré maximum bornés (alors qu'il existe un PTAS dans le cas non orienté).

Enfin, nos derniers résultats concernent le problème FMTEM dans les graphes orientés. Nous donnons les premiers résultats de complexité et d'inapproximabilité pour ce problème en montrant un lien avec MFEM, puis nous nous intéressons à la conception d'algorithmes approchés et à l'étude de cas particuliers polynomiaux. Nous introduisons un paramètre $k_S \leq |\mathcal{T}|$[1] pour ce problème, dont nous détaillons les propriétés, puis nous donnons une caractérisation complète, vis-à-vis de ce paramètre, des cas polynomiaux ou $\mathcal{NP}$-difficiles pour FMTEM. En particulier, nous exhibons un cas polynomial généralisant le cas des GSC [52]. Enfin, nous donnons un algorithme $2\log_2(k_S+2)$-approché pour ce problème, améliorant (pour $|\mathcal{T}| > k_S+2$) le précédent ratio de $2\log_2|\mathcal{T}|$ obtenu par Garg et al. [78], et nous mon-

[1] k_S est le nombre de terminaux *solitaires*, qui forment un sous-ensemble des terminaux.

trons, à l'aide d'une famille d'exemples, que notre analyse (ainsi que celle de Garg et al.) est *fine*. Ces résultats concernant FMTEM ont fait l'objet d'une publication en revue [17].

Organisation du présent document

Nos résultats peuvent se répartir en trois grandes parties.

Dans la première partie, nous étudions MFEM et MCM dans des graphes de largeur d'arbre bornée. Le premier chapitre de cette partie est consacré à la résolution de MFEM et MCDA dans les graphes de nombre cyclomatique borné et dans les graphes planaires à k niveaux d'arêtes. Dans le second chapitre, nous étudions la complexité de MCM dans les graphes (orientés ou non) de largeur d'arbre bornée. Un bref troisième chapitre détaille nos quelques résultats concernant MFEM dans les anneaux.

Dans la deuxième partie, nous étudions MFEM et MCM dans les graphes planaires. Le premier chapitre de cette partie est consacré au cas des grilles. Dans le second chapitre, nous montrons que l'algorithme de Yeh pour résoudre CMTM lorsque le nombre de liaisons est fixé est erroné, puis nous décrivons un algorithme polynomial résolvant MCM dans les graphes planaires lorsque les terminaux sont sur la face extérieure et en nombre fixé.

Enfin, la troisième partie traite des problèmes de chemins dans les graphes généraux. Dans le premier chapitre de cette partie, nous faisons une comparaison expérimentale de deux heuristiques très simples utilisées pour résoudre MFEM : la première est basée sur SPF, la seconde sur l'un des algorithmes décrits dans notre première partie. Nous étudions les premiers résultats bruts, et montrons ensuite comment les améliorer en combinant les deux méthodes, pour parvenir à un gain de temps notable. Le second chapitre concerne la complexité et l'approximation de FMTEM dans les graphes orientés.

Un glossaire est donné en annexe B, et l'ensemble de nos résultats est récapitulé en annexe C.1.

Première partie

Graphes de largeur d'arbre bornée

Chapitre 2

Multiflots entiers et chemins disjoints dans des graphes de largeur d'arbre bornée

2.1 Préliminaires

Dans ce chapitre, nous étudions MFEM et MCDA dans des graphes non orientés de largeur d'arbre bornée. Rappelons que Garg et al. ont montré un certain nombre de résultats forts sur ces deux problèmes dans les arbres [80]. Nous les détaillons à nouveau, car ils nous seront utiles dans ce chapitre. Tout d'abord, ces auteurs ont montré que MCDA était polynomial dans les arbres (c'est d'ailleurs l'une des rares classes de graphes, avec les anneaux, où ce problème est polynomial), et que MFEM y était APX-difficile (même si toutes les capacités valent 1 ou 2). Ensuite, ils ont donné un algorithme 2-approché pour MFEM (et MCM, puisqu'il suit une approche primale-duale), que nous décrivons plus bas, car nous en ferons une (ré)utilisation intensive. Leur résultat implique aussi une borne supérieure de 2 sur le ratio d'intégrité pour MFEM (et MCM) dans les arbres. Enfin, toujours dans [80], ils ont montré qu'il existait une famille de graphes planaires avec tous les terminaux sur la face extérieure où le ratio d'intégrité pour MCDA est $\Omega(\sqrt{n})$ (une famille similaire est donnée dans la figure 2.1).

Remarquons que, dans cette famille d'instances, tous les sommets qui ne sont pas sur la face extérieure ont un degré impair (tous les degrés valent 3) : c'est nécessaire pour obtenir un ratio d'intégrité élevé, puisque Kleinberg a montré que, si tous les degrés sont pairs (sauf, éventuellement, ceux des sommets se trouvant sur la face extérieure), le ratio d'intégrité est $O(\log^2 n)$ (et il existe un algorithme $O(\log^2 n)$-approché) [105]. De même, si toutes les capacités sont supérieures ou égales à 2 (et non unitaires comme dans cet exemple), le ratio d'intégrité est $O(\log n)$ [35, 37]. Enfin, rappelons que la preuve de Garg et al. établissant l'APX-difficulté de MFEM dans les arbres

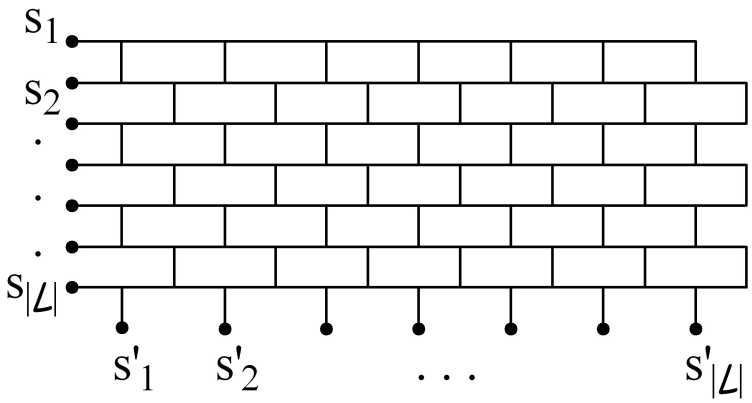


FIG. 2.1 – Une famille d'instances (dites graphes en *mur*) avec un ratio d'intégrité $\Omega(\sqrt{n})$ pour MCDA.

permet de montrer que MCDA est APX-difficile dans les cactus.

Nous montrons d'abord que *tous* les résultats obtenus par Garg et al. pour MCDA et MFEM dans les arbres [80] peuvent se généraliser aux graphes de nombre cyclomatique borné : plus précisément, MCDA y est polynomial, et MFEM admet un algorithme approché à ratio constant $2(\gamma + 1)$, où γ est le nombre cyclomatique ($\gamma = 0$ dans un arbre, et on retrouve donc le résultat de Garg et al.). En particulier, nous montrons que le ratio d'intégrité pour MFEM est borné par $2(\gamma+1)$, car nous utilisons une approche primale-duale.

Il convient de noter que d'autres généralisations des arbres ne peuvent pas mener à de tels résultats. Une façon de généraliser les arbres est de considérer les graphes de largeur d'arbre bornée. Cependant, rappelons que MCDA reste APX-difficile dans les cactus (des graphes de largeur d'arbre 2). En outre, le meilleur algorithme approché connu pour MFEM dans les graphes de largeur d'arbre bornée est un algorithme de ratio $O(\tau \log \tau \log n)$ non constant dû à Chekuri et al. (où τ est la largeur d'arbre) [41] ; ce résultat est postérieur à ceux présentés dans ce chapitre. Une deuxième façon de généraliser les arbres est de considérer des graphes où tous les terminaux sont sur un nombre borné de faces [42, 128]. Mais, dans ce cas, MCDA est également APX-difficile (les cactus appartenant à cette classe de graphes), et le ratio d'intégrité pour MFEM est $\Omega(\sqrt{n})$, même si le graphe est planaire et tous les terminaux sont sur la face extérieure (voir figure 2.1).

Nous montrons également, dans la suite de ce chapitre, que le ratio d'intégrité de MCDA est borné par $4k$ dans les **graphes planaires à k niveaux d'arêtes** (nous les définissons formellement dans la section suivante). Ces graphes, qui n'ont jamais été étudiés à notre connaissance, ont pourtant des propriétés intéressantes :

- Ils généralisent les cactus ;
- Ils forment une sous-classe naturelle des graphes planaires à k niveaux de sommets ;
- Ils ont une largeur d'arbre $\Theta(k)$, c.-à-d., bornée mais quelconque ;
- Ils incluent les graphes identiques à ceux de la figure 2.1 où l'une des deux dimensions est bornée.

Ces résultats n'ont pas été inclus dans la partie "Graphes planaires" de cette thèse, car la propriété la plus importante dans leur conception reste le fait que la largeur d'arbre de ces graphes est bornée (même si la planarité joue évidemment un rôle important) ; en particulier, le ratio d'approximation est $4k = \Theta(\tau)$. Remarquons qu'une démarche similaire à la nôtre a été utilisée récemment pour montrer un saut d'intégrité constant dans les graphes planaires à k niveaux pour un problème d'un autre type : il est conjecturé que ce problème a un saut d'intégrité constant dans les graphes planaires et dans les graphes de largeur d'arbre bornée, mais, avant les travaux présentés dans [38], seuls les cas des arbres et des graphes planaires superficiels avaient été traités. Les auteurs de [38] considèrent donc le cas des graphes planaires à k niveaux (où ils parviennent à montrer un ratio d'intégrité constant), qui offrent l'avantage de constituer une sous-classe naturelle des graphes planaires de largeur d'arbre bornée (mais quelconque, contrairement aux graphes séries-parallèles, par exemple). On peut noter à ce propos que si l'on spécialise l'algorithme de Chekuri et al. pour MFEM dans les graphes de largeur d'arbre bornée aux graphes planaires à k niveaux (ou même à k niveaux d'arêtes), on ne sait pas, pour l'instant, obtenir un ratio meilleur que $O(\tau \log n)$ (on gagne donc un facteur $\log \tau$ par rapport au cas général) ; notre problème semble donc encore plus difficile que celui considéré dans [38]. Enfin, précisons que, à notre connaissance, les résultats de ce chapitre, qui ont fait l'objet d'une publication en conférence internationale [15], forment la première tentative de généralisation du théorème de Garg et al. [80] établissant un ratio constant entre la valeur d'un multiflot entier maximum et la valeur d'une multicoupe minimum dans les arbres. En effet, nous verrons en section 2.1.2 qu'il existe des familles d'arbres où les algorithmes donnés dans [108] et [133], basés sur SPF, ne peuvent avoir un ratio d'approximation meilleur que $O(\sqrt{n})$.

Nous donnons à présent quelques définitions, idées et résultats préliminaires.

2.1.1 Définitions

Nous commençons par définir la classe des **graphes planaires à k niveaux d'arêtes** (que nous avons appelés *k-edge-outerplanar graphs* en anglais), qui a été inspirée par la classe des graphes planaires à k niveaux. Étant donné $k \geq 1$, un graphe planaire à k niveaux d'arêtes est un graphe

planaire ayant un plongement avec au plus k *niveaux* d'arêtes, c'est-à-dire, tel que, après avoir enlevé de façon itérative toutes les arêtes se trouvant sur la face extérieure au plus k fois, on obtient un graphe sans arêtes. Remarquons dès à présent que les graphes à 1 niveau d'arêtes (que nous avons appelés *edge-outerplanar* en anglais) sont les cactus. Nous détaillerons en section 2.3.1 les relations entre graphes planaires à k niveaux de sommets et graphes planaires à k niveaux d'arêtes. Notons qu'une grille planaire $2k \times N$ ($N > 2k$) est un graphe planaire à k niveaux de sommets et d'arêtes.

Nous définissons également la notion de **degrés intérieurs**. Étant donnés un graphe non orienté et une de ses composantes 2-sommet-connexes $2SC$, le degré d'un sommet v *à l'intérieur de* $2SC$, noté $deg_{2SC}(v)$, est le nombre de sommets de $2SC$ qui sont adjacents à v. Un sommet est de degré intérieur borné si son degré à l'intérieur de *chaque* composante 2-sommet-connexe est borné. Notons qu'un sommet peut avoir un degré intérieur borné et un degré non borné (l'inverse étant évidemment faux).

Enfin, nous définissons deux autres classes de graphes. Étant donnés deux nombres entiers $k \geq 1$ et $d \geq 2$, la classe des graphes planaires à k niveaux (de sommets) ayant un degré maximum borné par d à l'intérieur de chaque bloc sera désignée par $PN_k - DIB_d$. Étant donné un nombre entier $\gamma \geq 0$, la classe des graphes connexes $G = (S, A)$ ayant un nombre cyclomatique $\nu(G) = |A| - |S| + 1$ inférieur ou égal à γ sera désignée par NC_γ.

Notons que tous les graphes considérés dans ce chapitre sont non orientés, *simples* (c.-à-d., sans arêtes parallèles), *sans boucles* et *connexes* (si ce n'est pas le cas, nous considérons chaque composante connexe indépendamment).

Un algorithme 2-approché pour MFEM dans les arbres. Nous décrivons à présent l'algorithme 2-approché pour MFEM dans les arbres conçu par Garg et al. dans [80]. Il suit une approche primale-duale, et est donc également 2-approché pour MCM.

On commence par enraciner l'arbre T en un sommet quelconque r. Puis, dans une première phase, on parcourt l'arborescence obtenue, palier par palier, en partant du plus *haut* (c'est-à-dire celui de plus grande hauteur). Pour chaque palier et pour chacun de ses sommets v, on fait : pour chaque liaison (s_i, s'_i) non encore considérée telle que v est sur le chemin de r à s_i et sur celui de r à s'_i, router le plus d'unités de flots possible de s_i à s'_i. À la fin de cette phase (c'est-à-dire quand on atteint r), on obtient le multiflot entier F_T. On construit ensuite une multicoupe, en sélectionnant toutes les arêtes saturées. La deuxième phase consiste à obtenir la multicoupe C_T en parcourant l'arborescence en sens inverse (de la racine vers les feuilles) et en éliminant progressivement des arêtes redondantes parmi les arêtes saturées, de telle façon qu'au plus deux arêtes saturées de chaque chemin de routage soient coupées (en fait, pour un chemin de routage de s_i à s'_i, il y a au plus une arête coupée sur le chemin de s_i à v et sur le chemin de v à s'_i, où v est

le sommet le plus "haut" appartenant au chemin de s_i à s'_i ; ceci permet aussi de montrer, en particulier, l'optimalité de cet algorithme lorsque l'arbre est une arborescence, car, dans ce cas, $v = s_i$). Ainsi, Garg et al. montrent le théorème suivant :

Théorème 2.1 ([80]). $\|C_T\| \leq 2\|F_T\|$. *En outre, C_T et F_T peuvent être calculés en temps polynomial.*

2.1.2 Un algorithme approché simple pour MCDA

Rappelons que le ratio d'approximation de l'algorithme glouton SPF est $O(\sqrt{m})$ (voir la section 1.3.3). En outre, il existe des familles d'arbres où cette borne est atteinte. Notre premier résultat de ce chapitre est de donner une telle famille ici. D'abord, on commence par considérer une chaîne $v_1, \ldots, v_{p+2}$ de longueur $p+1$. Puis, on ajoute une chaîne $v_i, \ldots, s_i$ de longueur $p+1$ pour chaque $i \in \{2, \ldots, p+1\}$. On place ensuite s_1 sur v_1, s'_1 sur v_{p+2} et s'_i sur v_{i+1} pour chaque $i \in \{2, \ldots, p+1\}$. Ce graphe a $\Theta(p^2)$ sommets et arêtes (c.-à-d., $p = \Theta(\sqrt{n})$) et $p+1$ liaisons (c.-à-d., $|\mathcal{L}| = p+1$), et la chaîne de s_i à s'_i a une longueur de $p+2$, pour chaque $i \in \{2, \ldots, p+1\}$. SPF route (s_1, s'_1) (qui a une longueur de $p+1$), alors que la solution optimale est de router $(s_2, s'_2), \ldots, (s_{p+1}, s'_{p+1})$. Enfin, le graphe est un arbre et le graphe obtenu en ajoutant les $p+1$ arêtes (s_i, s'_i) reste planaire.

Par conséquent, même pour des classes très particulières de graphes, de meilleurs algorithmes approchés seraient nécessaires. Étant donné un graphe connexe non orienté G, plusieurs de nos résultats reposent sur la même idée de base : calculer un arbre couvrant de G afin d'utiliser les résultats donnés dans [80] pour les arbres. Puisque nous utiliserons plusieurs fois un nouvel algorithme simple fondé sur cette idée, nous le donnons ici. Nous le noterons $ACGT$ (pour *calcul d'un Arbre Couvrant, application de l'algorithme de Garg et al., et construction d'une multicoupe pour la Totalité du graphe*), et il peut être vu comme un schéma primal-dual contenant trois étapes :

1. Calculer un arbre couvrant T de G ;
2. À l'aide de l'algorithme donné dans [80], construire un multiflot entier F_T et une multicoupe C_T pour T tels que $\|C_T\| \leq 2\|F_T\|$;
3. Construire une multicoupe C_G pour G telle que $\|C_G\| \leq \alpha\|C_T\|$ pour un $\alpha > 0$ fixé.

À la fin de cet algorithme, on obtient un multiflot entier F_T et une multicoupe C_G tels que $\|C_G\| \leq 2\alpha\|F_T\|$. Comme F_T est également admissible pour G, cet algorithme est donc un algorithme 2α-approché à la fois pour MFEM et MCM. Évidemment, la première étape peut nécessiter un certain soin, et la troisième également (même si notre objectif n'est pas de trouver le meilleur α). Notons que l'étape 3 n'est utile que pour prouver le ratio d'approximation : si on est seulement intéressé par calculer un multiflot entier

approché, seules les deux premières étapes sont nécessaires. En outre, ces résultats sont nettement moins intéressants pour MCM que pour MFEM, car il est déjà connu pour MCM que le ratio d'intégrité est $O(1)$ (et qu'il existe un algorithme $O(1)$-approché) dans les graphes planaires et dans les graphes de largeur d'arbre bornée [157].

2.2 Graphes avec un nombre cyclomatique borné

Dans cette section, on généralise les résultats obtenus pour les arbres dans [80] aux graphes de NC_γ. Nous montrons que MCDA peut être résolu en temps polynomial pour les graphes dans NC_γ, en résolvant $O((2^\gamma|\mathcal{L}| + 1)^\gamma)$ instances définies sur des ensembles d'arbres. Nous montrons également comment appliquer ces idées afin d'obtenir un algorithme approché pour MCDA dans les *graphes quasi-complets*. Puis, étant donné un graphe G dans NC_γ, nous montrons comment calculer un multiflot entier F_G et une multicoupe C_G tels que $\|C_G\| \leq 2(\gamma + 1)\|F_G\|$, en utilisant $ACGT$. Enfin, nous montrons que MCM peut être résolue en temps $O(m^{2^\gamma|\mathcal{L}|})$ pour les graphes dans NC_γ, ce qui est polynomial en m si $|\mathcal{L}|$ est fixé.

2.2.1 Résoudre MCDA

Garg et al. ont prouvé que MCDA est polynomial dans les arbres. Nous utilisons à présent ce résultat pour concevoir un algorithme polynomial pour résoudre MCDA dans les graphes de NC_γ.

Soit G un graphe de NC_γ. On enlève γ arêtes de G, de sorte que le graphe obtenu soit un arbre couvrant, en sélectionnant de façon itérative une arête située dans un bloc. Soient $a_1, \ldots, a_\gamma$ ces arêtes. L'idée principale est que, puisque γ est fixé, il y a un nombre limité d'arêtes qui doivent être considérées. Pour chacune de ces γ arêtes, on choisit une chaîne élémentaire de P qui la traverse (ou aucune chaîne), et on enlève cette arête ainsi que toutes les autres arêtes traversées par la chaîne choisie (si elle est définie). On doit bien faire attention à ne choisir que des chaînes compatibles (c'est-à-dire, disjointes par les arêtes) : par exemple, si on choisit une chaîne p traversant a_i, on doit également choisir p pour a_j, $i \neq j$, si p traverse a_j. Après avoir fait cela pour les γ arêtes, on obtient une forêt. On calcule alors une solution optimale pour MCDA sur cette forêt en utilisant l'algorithme de Garg et al. [80]. En réunissant l'ensemble des chaînes sélectionnées dans cette solution avec celles choisies précédemment, on obtient une solution pour MCDA dans G. On répète cette procédure jusqu'à ce que chaque combinaison possible des chaînes élémentaires traversant $a_1, \ldots, a_\gamma$ ait été essayée (rappelons que, en fait, pour chacune de ces γ arêtes, on doit également essayer le cas où aucune chaîne ne l'emprunte). En gardant la meilleure de toutes ces solutions, on obtient la solution optimale.

Notre algorithme résout $O((|P|+1)^\gamma)$ instances de MCDA sur des forêts. Donc, γ étant fixé, si $|P|$ est polynomial en n et $|\mathcal{L}|$, notre algorithme s'exécute en temps polynomial. Le lemme suivant montre une borne sur $|P_i|$ pour chaque i [1] :

Lemme 2.1. *Étant donné un graphe G de NC_γ et deux sommets s_i et s'_i, le nombre de chaînes élémentaires $|P_i|$ reliant s_i à s'_i dans G est au plus 2^γ.*

Démonstration. On procède par récurrence sur γ. Pour $\gamma = 0$, on a $|P_i| = 1$ (G est un arbre). Supposons alors que la propriété soit vraie pour $\gamma-1$, $\gamma \geq 1$, et montrons que cela implique qu'elle est vraie pour γ. Si $|P_i| = 1$, on a fini. Sinon, soit v le premier sommet rencontré dans toute chaîne élémentaire de s_i à s'_i qui se trouve dans un bloc. Nous pouvons supposer s.p.d.g. que $v = s_i$ (si ce n'est pas le cas, cette hypothèse ne modifie pas $|P_i|$). Ainsi, il existe au moins deux arêtes, a_1 et a_2, adjacentes à s_i et se trouvant dans un bloc. Aucune chaîne élémentaire de s_i à s'_i ne traverse à la fois a_1 et a_2 : par conséquent, il existe une arête $a \in \{a_1, a_2\}$ telle qu'au moins la moitié des chaînes de P_i ne traverse pas a. En outre, si on enlève a, on obtient un graphe $G' \in NC_{\gamma-1}$, et on peut alors appliquer l'hypothèse de récurrence : il y a au plus $2^{\gamma-1}$ chaînes élémentaires entre s_i et s'_i dans G'. Par conséquent, $|P_i| \leq 2 \cdot 2^{\gamma-1} = 2^\gamma$. Le lemme est donc démontré. □

Notons que cette borne est atteinte : pour le voir, on peut, par exemple, considérer une chaîne de longueur γ ayant s_i et s'_i comme extrémités. Si on remplace chaque arête (u, v) de cette chaîne par un cycle (u, w, v, w', u), on obtient un graphe dans NC_γ qui vérifie $|P_i| = 2^\gamma$. En outre, le lemme 2.1 implique que $|P| \leq 2^\gamma|\mathcal{L}|$, et donc l'algorithme donné ci-dessus s'exécute en temps polynomial. Par conséquent :

Théorème 2.2. *MCDA est polynomial pour les graphes dans NC_γ.*

Remarquons que notre algorithme est FPT vis-à-vis du couple de paramètres $(|\mathcal{L}|, \gamma)$. Il reste cependant à déterminer s'il existe un algorithme FPT pour MCDA, si le seul paramètre est le nombre cyclomatique (notre algorithme n'étant clairement pas FPT dans ce cas). En outre, rappelons que le problème MCDA est APX-difficile même pour $|\mathcal{L}| = 2$ [89]. Néanmoins, en utilisant les résultats de cette section, on a :

Théorème 2.3. *Si $|\mathcal{L}|$ est fixé, MCDA est polynomial dans les graphes dont le nombre cyclomatique est $O(\sqrt{\log n})$.*

Démonstration. Nous utilisons l'algorithme ci-dessus. Rappelons que l'on a à résoudre $O((2^\gamma|\mathcal{L}|+1)^\gamma)$ instances de MCDA dans des forêts, où γ est le nombre cyclomatique. Donc, si $\gamma = O(\sqrt{\log n})$ et si $|\mathcal{L}|$ est fixé, nous avons à résoudre $O(n^{O(1)})$ instances, ce qui est polynomial en n. □

[1] Nous n'avons pas été en mesure de déterminer si ce résultat était déjà connu. Quoi qu'il en soit, nous fournissons une courte preuve, par souci de complétude.

Notons que les théorèmes 2.2 et 2.3 (et leurs analyses) sont également valides pour MCDS (où l'on cherche des chemins disjoints par les sommets, et non plus par les arêtes), puisque ce problème est également polynomial dans les arbres [27]. En utilisant des idées similaires (c.-à-d., en ajoutant ou en enlevant un nombre *constant* d'arêtes), on peut également dériver de [28] et de [35, 37] les résultats suivants :

Proposition 2.1. *Il existe un algorithme* $9(\beta+1)$*-approché pour MLCDA dans les graphes* $G = (S, A)$ *où* $|A| = \frac{|S|(|S|-1)}{2} - \beta$ *(ou* graphes quasi-complets*).*

Démonstration. On ajoute β arêtes à G pour le transformer en un graphe complet G', puis on calcule une solution pour MLCDA dans G' en utilisant l'algorithme donné dans [28]. Ensuite, on enlève les β arêtes qui ont été ajoutées à G et les chaînes de la solution qui les traversent (il y a au plus β chaînes de ce type). S'il y avait au moins $\beta+1$ chaînes routées dans G', on a fini. Sinon, on route n'importe quelle chaîne dans G et on perd un facteur au plus β dans la valeur de la solution. Dans les deux cas, on obtient le ratio désiré, puisque l'algorithme dans [28] est un algorithme 9-approché pour MLCDA dans les graphes complets. □

Proposition 2.2. *Il existe un algorithme* $O(\log n)$*-approché pour MFEM dans les graphes planaires ayant* $\beta = O(\log n)$ *arêtes de capacité* 1*.*

Démonstration. On enlève les β arêtes de capacité 1, puis on utilise l'algorithme de Chekuri et al. [35, 37] sur le graphe obtenu. Si cet algorithme ne renvoie pas une solution de valeur 0, alors on obtient une solution de valeur au moins $1/(O(\log n) + \beta) = 1/O(\log n)$ fois la valeur d'une solution optimale, et la proposition est démontrée. Sinon, on route n'importe quelle chaîne dans le graphe initial, et on obtient un algorithme β-approché. □

Enfin, il est intéressant de remarquer que le lemme 2.1 permet de montrer que MCM est polynomial pour les graphes dans NC_γ, si $|\mathcal{L}|$ est fixé ; la preuve est très similaire à celle de Garg et al. pour le cas des arbres [80]. Il suffit de remarquer que toute multicoupe optimale contient au plus $|\mathcal{L}|2^\gamma$ arêtes, car $|P_i| \leq 2^\gamma$ pour tout i. Donc, en énumérant tous les ensembles d'arêtes de taille au plus $|\mathcal{L}|2^\gamma$ et en gardant celui de plus faible poids, on obtient une multicoupe minimum. Comme cette taille est fixée ($|\mathcal{L}|$ et γ l'étant), on peut faire cette énumération en temps polynomial. Cependant, nous montrons en section 3.2 qu'une autre approche permet de montrer un résultat plus général : si $|\mathcal{L}|$ est fixé, MCM est polynomial dans les graphes de largeur d'arbre bornée.

2.2.2 Borner le ratio d'intégrité pour MFEM dans NC_γ

Dans cette section, nous montrons comment, étant donné un graphe G de NC_γ, calculer en temps polynomial à l'aide de l'algorithme $ACGT$ un

multiflot entier F_G et une multicoupe C_G tels que $\|C_G\| \leq 2(\gamma+1)\|F_G\|$.

Nous n'avons, pour cela, qu'à détailler comment construire un arbre couvrant T pour G (étape 1), et ensuite, comment construire une multicoupe C_G telle que $\|C_G\| \leq (\gamma+1)\|C_T\|$ (étape 3).

L'étape 1 se déroule comme suit : on construit un arbre couvrant de poids maximum de G, en utilisant une variante de l'algorithme de Kruskal [115]. En d'autres termes, on sélectionne de façon itérative une arête a_i ayant la capacité minimum parmi toutes celles se trouvant dans un bloc de $G \setminus \{a_1, \ldots, a_{i-1}\}$. Ceci nous donne un ensemble de γ arêtes vérifiant $c(a_1) \leq c(a_2) \leq \cdots \leq c(a_\gamma)$, et le graphe $G \setminus \bigcup_{i\in\{1,\ldots,\gamma\}} a_i$ est un arbre, T.

Dans l'étape 2, on calcule pour T un multiflot entier $F_T = F_G$ et une multicoupe C_T tels que $\|C_T\| \leq 2\|F_T\|$. Ensuite, dans l'étape 3, on utilise C_T pour construire une multicoupe C_G pour G. Pour chaque arête b_j de C_T qui se trouve dans un bloc de G, on désigne par $\lambda(b_j)$ le plus grand i tel que, juste avant que l'arête a_i ait été enlevée de $G \setminus \{a_1, \ldots, a_{i-1}\}$, b_j se trouvait toujours dans un bloc. En outre, soit $\lambda^* = \max_{b_j \in C_T} \lambda(b_j)$ et soit $b_{j^*} \in C_T$ l'arête telle que $\lambda(b_{j^*}) = \lambda^*$. Alors, on définit $C_G = C_T \cup \bigcup_{i\in\{1,\ldots,\lambda^*\}} \{a_i\}$.

Montrons d'abord que $\|C_G\| \leq (\gamma+1)\|C_T\|$. On a $\|C_G\| = \|C_T\| + \sum_{i=1}^{\lambda^*} c(a_i) \leq \|C_T\| + \lambda^* c(a_{\lambda^*}) \leq \|C_T\| + \lambda^* c(b_{j^*}) \leq \|C_T\| + \lambda^*\|C_T\| = (\lambda^*+1)\|C_T\|$. Notons que l'inégalité $c(a_{\lambda^*}) \leq c(b_{j^*})$ vient des définitions de λ^* et b_{j^*}, et de la manière dont a_{λ^*} a été choisie.

Ensuite, prouvons que C_G est bien une multicoupe pour G. En fait, tout ce que nous avons à prouver est que, étant donnée une arête appartenant à $\{a_{\lambda^*+1}, \ldots, a_\gamma\}$, il est inutile de la sélectionner dans C_G, c'est-à-dire, il existe dans $T \setminus C_T$ une autre chaîne reliant ses deux extrémités. Soit (u, v) une arête appartenant à $\{a_{\lambda^*+1}, \ldots, a_\gamma\}$. Il existe une chaîne entre u et v dans T, puisque T est un arbre couvrant de G. Donc, s'il n'existe aucune chaîne de u à v dans $T \setminus C_T$, alors nécessairement la chaîne de u à v dans T contient une arête b appartenant à C_T. Ceci implique que, juste avant que (u, v) ait été enlevée, b se trouvait dans un bloc. Puisque $(u, v) \in \{a_{\lambda^*+1}, \ldots, a_\gamma\}$, on obtient une contradiction. On a $\lambda^* \leq \gamma$, et donc :

Théorème 2.4. *Le ratio entre les optimums de MCM et MFEM est borné par* $2(\gamma+1)$ *pour les graphes dans* NC_γ*. En outre, des solutions pour MCM et MFEM réalisant ce rapport peuvent être calculées en temps polynomial.*

Corollaire 2.1. *Le ratio d'intégrité pour MFEM est borné par* $2(\gamma+1)$ *pour les graphes dans* NC_γ*. En outre, une solution pour MFEM réalisant ce rapport peut être calculée en temps polynomial.*

Notons que ces résultats s'appliquent aussi à MFEMI, puisque la solution calculée par notre méthode est admissible pour ce problème. Notons également que, dans l'analyse du théorème 2.4, savoir explicitement que l'arbre couvrant construit dans l'étape 1 est en fait un arbre couvrant de poids maximum n'est pas nécessaire (et savoir comment il est construit suffit). En

outre, nous ne savons pas si la borne $2(\gamma + 1)$ est fine ou non (évidemment, c'est le cas pour $\gamma = 0$ [80]). La figure 2.2 montre un exemple où une borne plus faible est atteinte.

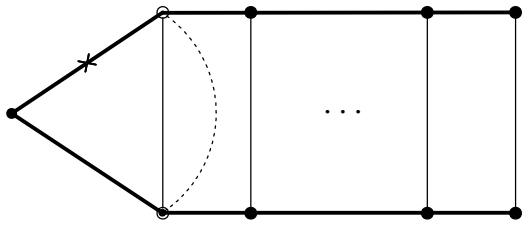


FIG. 2.2 – Un exemple pour le théorème 2.4 avec une liaison (en pointillés). Les arêtes en traits gras (c.-à-d., les arêtes formant l'arbre couvrant) ont une capacité de $N + 1$ pour un entier $N > 0$, alors que toutes les autres arêtes ont une capacité de N. Donc, $\|F_T\| = N + 1$, $\|C_T\| = N + 1$ (arête désignée par $\ast$) et $\|C_G\| = \gamma N + (N + 1)$.

Enfin, on peut remarquer que, dans les graphes où $\max_{a \in A} c(a) \leq \beta$ pour un $\beta \in \mathbb{N}^*$, si on utilise notre algorithme en posant $C_G = C_T \cup \bigcup_{i \in \{1,\ldots,\gamma\}} \{a_i\}$, on obtient une multicoupe C_G et un multiflot entier F_T tels que $\|C_G\| \leq \|C_T\| + \gamma\beta \leq 2\|F_T\| + \gamma\beta = 2(1 + o(1))\|F_T\|$. Cela fournit une généralisation du théorème de Garg et al. pour les arbres, différente de celle du théorème 2.4, mais ne s'appliquant qu'aux graphes de NC_γ ayant des capacités bornées.

2.3 Ratios d'intégrité dans les graphes planaires à k niveaux d'arêtes

Dans cette section, nous étudions le cas des graphes planaires à k niveaux d'arêtes. Nous prouvons d'abord que les graphes planaires à k' niveaux ayant un degré intérieur borné par d (c'est-à-dire les graphes dans $PN_{k'} - DIB_d$) ont un lien étroit avec ces graphes.

2.3.1 Relation entre graphes planaires à k niveaux de sommets et graphes planaires à k niveaux d'arêtes

Le résultat principal de cette section est donné dans le théorème 2.5 :

Théorème 2.5. *Tout graphe planaire à k niveaux tel que le degré de chaque sommet est borné par $d \geq 2$ à l'intérieur de chaque bloc est un graphe planaire à $(\lceil \frac{d}{2} \rceil + (k-1)\lfloor \frac{d}{2} \rfloor)$ niveaux d'arêtes. En outre, tout graphe planaire à k niveaux d'arêtes est un graphe planaire à k niveaux.*

Démonstration. La deuxième partie du théorème 2.5 est évidente. Nous prouvons la première partie par récurrence. Soit G un graphe dans $PN_k - DIB_d$,

$k \geq 2$. Notons que, pour la preuve, nous pouvons considérer chaque bloc indépendamment. Soit $2SC$ une composante 2-sommet-connexe de G quelconque, maximale au sens de l'inclusion. Chaque sommet de $2SC$ se trouvant sur la face extérieure est adjacent à exactement deux arêtes de $2SC$ se trouvant sur la face extérieure. Pour chaque sommet de ce type, on enlève les deux arêtes correspondantes. En outre, on enlève toute arête qui ne se trouve pas dans un bloc. On répète ceci jusqu'à ce que chaque sommet de $2SC$ se trouvant sur la face extérieure de G ait au plus un voisin parmi les sommets qui sont dans $2SC$. À chaque itération, pour chaque sommet v se trouvant sur la face extérieure et ayant toujours au moins deux voisins parmi les sommets de $2SC$, on enlève deux arêtes adjacentes à v, donc nous devons le faire au plus $\frac{d}{2}$ fois si d est pair. Si d est impair, alors on s'arrête lorsque $deg_{2SC}(v)$ dans le graphe résiduel est au plus un, donc on doit le faire au plus $\frac{d-1}{2}$ fois, c.-à-d., au plus $\lfloor \frac{d}{2} \rfloor$ fois. Après cela, on obtient une composante dans $PN_{k-1} - DIB_d$. Ensuite, pour un graphe dans $PN_1 - DIB_d$, on utilise la même technique. Si d est pair, alors l'analyse est semblable. Si d est impair, alors, pour chaque sommet v, on doit rendre $deg_{2SC}(v)$ dans le graphe résiduel égal à 0, donc on doit enlever des arêtes sur la face extérieure $\lceil \frac{d}{2} \rceil$ fois. En conclusion, tout graphe dans $PN_k - DIB_d$ est un graphe planaire à $((k-1)\lfloor \frac{d}{2} \rfloor + \lceil \frac{d}{2} \rceil)$ niveaux d'arêtes. □

Ce théorème prouve que, pour qu'un graphe soit planaire à k niveaux d'arêtes pour un certain k, il est suffisant qu'il soit dans $PN_{k'} - DIB_d$ pour un certain k' et un certain d. Cependant, ce n'est pas une condition nécessaire (chaque graphe d'*Halin*, c.-à-d., chaque graphe planaire sans sommets de degré 2 et dont les arêtes sont faites de l'union disjointe d'un arbre et d'un cycle reliant les feuilles de cet arbre, est planaire à 2 niveaux de sommets et à 2 niveaux d'arêtes), et être seulement planaire à k' niveaux pour un certain k' n'est pas suffisant en général (pour tout $p > 2$, le graphe complet biparti $K_{2,p}$ est un graphe planaire à $\lceil \frac{p}{2} \rceil$ niveaux d'arêtes et 2 niveaux de sommets, le premier niveau ayant 4 sommets et le second $p-2$).

En outre, la figure 2.3 prouve que la borne du théorème 2.5 est fine. Dans la suite de la section 2.3, nous considérerons seulement des graphes planaires à k niveaux d'arêtes. Le théorème 2.5 prouve que nos résultats s'appliqueront, en particulier, aux graphes dans $PN_{k'} - DIB_d$.

2.3.2 Borner le saut d'intégrité pour MCDA

Rappelons que MCDA est $\mathcal{NP}$-difficile et APX-difficile dans les cactus [80]. Le résultat principal de cette section est que l'on peut borner le ratio d'intégrité pour MCDA dans les graphes planaires à k niveaux d'arêtes :

Théorème 2.6. *Le ratio d'intégrité pour* MCDA *est borné par* $4k$ *dans les graphes planaires à* k *niveaux d'arêtes. En outre, une solution pour* MCDA *réalisant ce rapport peut être calculée en temps polynomial.*

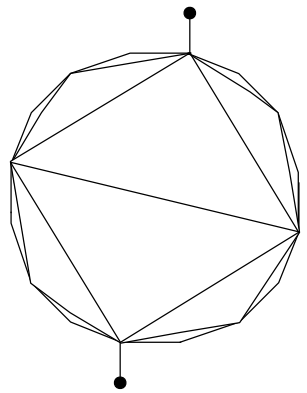


FIG. 2.3 – G_1, le squelette d'une famille de graphes atteignant la borne du théorème 2.5 (d impair). Chaque graphe G_i, $i \geq 2$, est en fait obtenu à partir de G_1 en remplaçant chaque arête par une copie de G_{i-1}, les deux gros sommets correspondant aux extrémités de cette arête. Pour un entier $k > 0$, le graphe planaire G_k a k niveaux de sommets et $((k-1)\lfloor \frac{d}{2} \rfloor + \lceil \frac{d}{2} \rceil)$ niveaux d'arêtes ($d = 7$ ici).

Démonstration. Nous utilisons l'algorithme $ACGT$ (voir la section 2.1.2). Décrivons les étapes 1, 2 et 3. Étant donné un graphe connexe $G = (S, A)$, planaire à k niveaux d'arêtes, l'étape 1 se déroule comme suit : (i) pour i=k à 1, (ii) pour chaque face interne Φ, s'il existe des arêtes se trouvant sur le i^e niveau d'arêtes de G et sur la frontière de Φ, on enlève exactement une telle arête. Après la fin de la phase (ii) au i^e niveau ($i \geq 2$), on obtient un graphe connexe, planaire à $(i-1)$ niveaux d'arêtes. Par conséquent, à la fin de l'étape 1 (c.-à-d., quand la phase (i) finit), on obtient un arbre couvrant T de G. Puis, F_T et C_T sont obtenus à l'étape 2. Ensuite, on utilise C_T pour construire C_G dans l'étape 3.

Pour chaque arête dans C_T, C_G contiendra au plus $2k$ arêtes, et donc $\|C_G\| \leq 2k\|C_T\| \leq 4k\|F_T\|$. La suppression de toute arête $(u, v) \in C_T$ sépare les sommets de T (et par conséquent ceux de G) en exactement deux composantes, S_u et $S_v = S \setminus S_u$. Soit $\delta_G(u, v)$ l'ensemble des arêtes situées entre S_u et S_v dans G. Le lemme suivant nous sera utile pour la suite :

Lemme 2.2. *Étant données S_u et S_v dans un graphe planaire à k niveaux d'arêtes G, $\delta_G(u, v)$ contient au plus 2 arêtes sur chacun des k niveaux. En outre, il contient exactement 2 arêtes sur le k^e niveau ssi elles sont sur le cycle extérieur de G.*

Démonstration. On procède par récurrence sur k. Pour $k = 1$, on a un cactus. Si (u, v) ne se trouve pas dans un cycle, alors $\delta_G(u, v)$ contient seulement (u, v) et on a fini. Sinon, d'après la façon dont nous construisons T, il y a

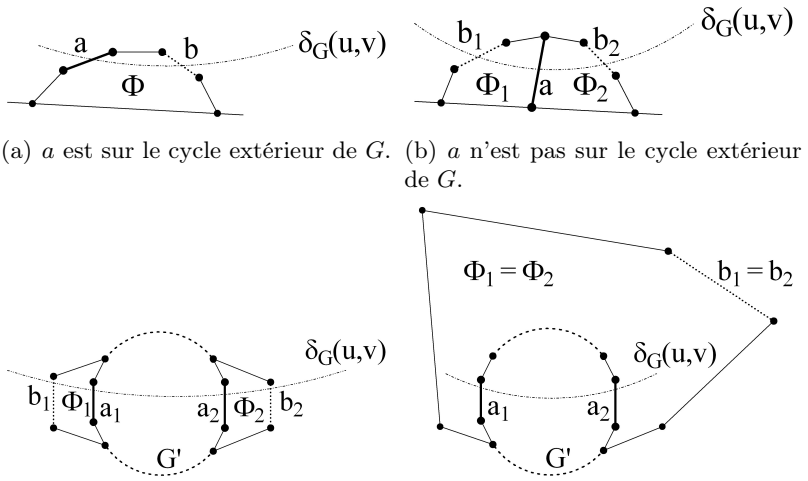
un cycle de G contenant (u, v) et une arête qui n'est pas dans T : $\delta_G(u, v)$ contient ces 2 arêtes. Ceci termine le cas $k = 1$.

Supposons maintenant que le lemme 2.2 soit vrai pour $k - 1$, $k \geq 2$, et considérons un graphe connexe, planaire à k niveaux d'arêtes. Pour chaque face interne ayant des arêtes en commun avec la face extérieure, une de ces arêtes a été enlevée pendant l'étape 1 ; enlevons-la de G à nouveau. Le graphe connexe obtenu G' est planaire à $(k-1)$ niveaux d'arêtes, donc nous pouvons appliquer l'hypothèse de récurrence. En outre, $\delta_{G'}(u, v) \subseteq \delta_G(u, v)$. Nous devons distinguer trois cas :

- S'il n'y a aucune arête sur le $(k-1)^e$ niveau de G' qui appartient à $\delta_{G'}(u, v)$, alors, évidemment, pour chaque arête a sur le cycle extérieur de G, il y a une chaîne reliant ses deux extrémités et utilisant uniquement des arêtes situées sur le $(k-1)^e$ niveau de G'. Par conséquent, a n'appartient pas à $\delta_G(u, v)$.
- S'il y a une arête (disons, a) sur le $(k-1)^e$ niveau de G' qui appartient à $\delta_{G'}(u, v)$, alors, par hypothèse, a n'est pas sur le cycle extérieur de G'. Supposons d'abord que a se trouve dans un bloc de G. Si a est sur le cycle extérieur de G, alors il y a une face interne Φ de G (adjacente à la face extérieure de G) dont la frontière contient a et une arête b n'étant pas dans G', les deux étant dans $\delta_G(u, v)$ (voir la figure 2.4(a)). Sinon, a appartient à la frontière de deux faces internes de G, Φ_1 et Φ_2. Φ_1 (resp. Φ_2) est adjacente à la face extérieure et sa frontière contient une arête b_1 (resp. b_2) qui n'est pas dans G', et qui appartient à $\delta_G(u, v)$ (voir la figure 2.4(b)). Notons que si a ne se trouve pas dans un bloc de G, alors aucune arête du cycle extérieur de G n'appartient à $\delta_G(u, v)$ (c.-à-d., $\delta_{G'}(u, v) = \delta_G(u, v)$).
- S'il y a deux arêtes a_1 et a_2 sur le $(k-1)^e$ niveau de G' appartenant à $\delta_{G'}(u, v)$, alors, par hypothèse, elles appartiennent au cycle extérieur de G'. Par conséquent, a_1 (resp. a_2) appartient à la frontière d'une face interne Φ_1 (resp. Φ_2) de G, adjacente à la face extérieure de G et contenant une arête b_1 (resp. b_2) n'appartenant pas à G'. b_1 et b_2 sont distinctes (et dans $\delta_G(u, v)$) ssi Φ_1 et Φ_2 sont distinctes (voir les figures 2.4(c) et 2.4(d)).

La preuve du lemme 2.2 est à présent terminée. □

Nous appliquons le lemme 2.2 pour chaque arête dans C_T (autrement dit, on a $C_G = \bigcup_{(u,v) \in C_T} \{\delta_G(u, v)\}$), et ceci implique immédiatement $\|C_G\| \leq 2k\|C_T\| \leq 4k\|F_T\|$, comme annoncé. Notons que la raison pour laquelle l'étape 1 doit être soigneusement exécutée est que, si T n'est pas construit comme indiqué, on ne sera pas capable de borner $|\delta_G(u, v)|$. Par exemple, dans la figure 2.2, supposons que chaque arête soit pondérée par 1 et que l'arbre couvrant construit à l'étape 1 soit constitué des arêtes en gras. Alors, $\|C_T\| = 1$ bien que $\delta_G(u, v)$ contienne toutes les arêtes en trait fin, donc $\|C_G\|$ est arbitrairement grand devant $\|C_T\|$. □



(a) a est sur le cycle extérieur de G. (b) a n'est pas sur le cycle extérieur de G.

(c) Φ_1 et Φ_2 sont différentes. (d) Φ_1 et Φ_2 ne sont pas différentes.

FIG. 2.4 – Illustration des 4 cas principaux du lemme 2.2.

Notons que, dans les graphes où toutes les arêtes ont la même capacité, le théorème 2.6 s'applique également à MFEM et MFEMI (puisque au plus un chemin de routage est associé à chaque liaison et que seuls des chemins de routage disjoints par les arêtes sont utilisés). Plus généralement, on a le corollaire suivant :

Corollaire 2.2. *Le ratio d'intégrité pour MFEM et MFEMI est borné par $4\beta k$ pour tout graphe $G = (S, A)$ planaire à k niveaux d'arêtes vérifiant $\max_{a\in A} c(a) \leq \beta \min_{a\in A} c(a)$. En outre, des solutions réalisant ce rapport peuvent être calculées en temps polynomial.*

La dernière remarque à faire concernant notre analyse de l'algorithme est qu'elle est fine : il existe, en effet, des familles d'instances où la coupe C_G et le flot F_T renvoyés par $ACGT$ vérifient $\|C_G\| = 4k\|F_T\|$. Nous en donnons une ici. On va construire un graphe G à partir d'un nombre *impair* de copies du graphe complet biparti $K_{2,d}$ (d pair). Pour la i^e copie de $K_{2,d}$, on note v_i et t_i ses deux sommets de degré d (les d autres étant de degré 2). On identifie tous les v_i en un seul sommet, v (et on a donc $v_1 = v_2 = \cdots = v_i = \cdots = v$). En outre, on définit toutes les liaisons (t_i, t_j), $i < j$ (on a donc $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$). Ce graphe est planaire à $\frac{d}{2}$ niveaux d'arêtes (car d est pair). De surcroît, pour tout arbre couvrant T, $\|C_T\| = |\mathcal{T}| - 1$ et $\|F_T\| = \frac{|\mathcal{T}|-1}{2}$ (car $|\mathcal{T}|$ est impair). Enfin, dans chacune des $|\mathcal{T}|-1$ premières copies de $K_{2,d}$, une arête est dans C_T et d arêtes sont dans C_G. On a donc $\|C_G\| = d(|\mathcal{T}|-1) = 2d\|F_T\|$, ce qui nous donne le résultat attendu (G étant planaire à $\frac{d}{2}$ niveaux d'arêtes). La figure 2.5 montre un exemple où $d = 4$ et $|\mathcal{T}| = 3$.

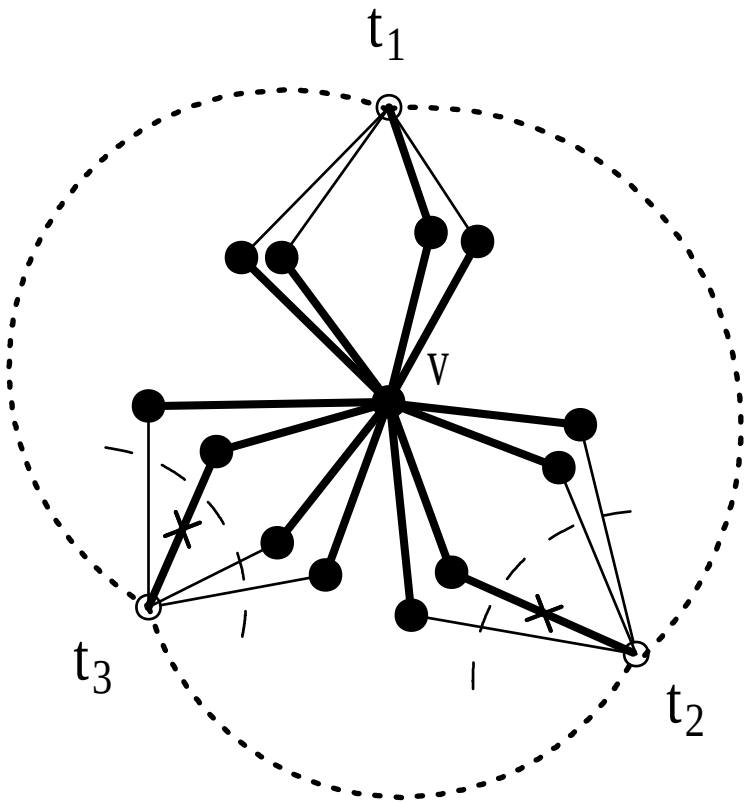


FIG. 2.5 – Un graphe planaire à deux niveaux d'arêtes, trois terminaux et où $d = 4$. Les trois liaisons sont en pointillés, et les arêtes de l'arbre couvrant T sont en gras. Ici, $\|F_T\| = 1$, $\|C_T\| = 2$ (arêtes désignées par $\times$) et $\|C_G\| = 8$ (arêtes traversées par les traits en pointillés).

Cependant, cela n'implique pas que le ratio dans l'énoncé du théorème 2.6 est fin. En effet, dans l'exemple ci-dessus, on a $Opt(MCDA) = 6$.

En outre, il faut noter que, malheureusement, notre approche échoue (et semble difficile à adapter) si la condition $\max_{a \in A} c(a) \leq \beta \min_{a \in A} c(a)$ n'est pas vérifiée. En effet, si l'on considère l'exemple donné en figure 2.2 et si l'on suppose que toutes les arêtes sur la face extérieure sont pondérées par un entier $N > 0$ et que toutes les autres arêtes sont pondérées par 1, l'approche décrite dans cette section fournit un multiflot entier F_T et deux multicoupes C_T et C_G tels que $\|F_T\| = 1$, $\|C_T\| = 1$ et $\|C_G\| = 2N + 1$ (en incluant toutes les arêtes de $\delta_G(u, v)$). Néanmoins, il faut remarquer que les conditions $c(a) \geq \beta$ (considérée dans [35, 37, 40, 140]) et $c(a) \leq \beta$ (considérée dans cette section et, à notre connaissance, pas dans les travaux précédents) pour tout $a \in A$ et pour un entier $\beta > 0$, ne sont pas du tout symétriques, la seconde étant nettement moins restrictive. En effet, la première ne permet pas d'inclure le problème MCDA (alors que, comme nous l'avons vu, la difficulté de MFEM est en grande partie capturée par celle de MCDA), et elle permet de réduire le saut d'intégrité à un facteur constant ou logarithmique (ce qui n'est évidemment pas le cas de la deuxième). Enfin, rappelons que la motivation initiale de l'étude de ces problèmes est la résolution de problèmes de coloration de chemins (voir la section 1.3.2), et que, sur ce point, la deuxième hypothèse est plus proche des conditions de problèmes réels dans les réseaux où, souvent, le nombre de longueurs d'onde est limité.

2.3.3 MFEM et MCM dans les cactus

Dans cette section, on considère la classe des graphes où le degré de chaque sommet est borné par deux (c.-à-d., est égal à 0 ou à 2) à l'intérieur de chaque bloc. Notons que c'est exactement la classe des graphes où deux composantes 2-sommet-connexes quelconques partagent au plus un sommet, c.-à-d., la classe des graphes où chaque bloc est un cycle (c.-à-d., un anneau) : par conséquent, c'est la classe des cactus.

La réduction polynomiale donnée dans [80] prouve que MCDA (et donc MFEM) est $\mathcal{NP}$-difficile et APX-difficile dans les cactus. En outre, Erlebach prouve que MLCDA est APX-difficile dans les arbres d'anneaux et donne un algorithme 3-approché pour ce cas [61]. Nous montrons à présent comment obtenir un algorithme 4-approché pour MFEM et MCM dans les cactus. L'idée est d'utiliser $ACGT$. Étant donné un cactus connexe G, on désigne par $\mathcal{C}_i$, $i \in \{1, \ldots, \rho\}$, son i^e cycle. Alors, pour chaque i, on enlève de $\mathcal{C}_i$ l'arête a_i ayant la plus petite capacité parmi toutes les arêtes de $\mathcal{C}_i$. De cette façon, on obtient T, un arbre couvrant de G de poids maximum, et on peut calculer un multiflot entier F_T et une multicoupe C_T pour T tels que $\|C_T\| \leq 2\|F_T\|$ en utilisant l'algorithme donné dans [80]. Ensuite, on construit une multicoupe C_G pour G : pour chaque cycle $\mathcal{C}_i$, on sélectionne a_i dans C_G si et seulement si il existe une autre arête de $\mathcal{C}_i$ dans C_T. On ajoute également à C_G toutes

les arêtes de C_T. On a $\|C_G\| = \|C_T\| + \sum_{a_i \text{ / il y a une arête de } \mathcal{C}_i \text{ dans } C_T} c(a_i) \leq 2\|C_T\| \leq 4\|F_T\|$. On voit facilement que C_G est bien une multicoupe pour G, puisque, pour chaque arête $a_i = (u_i, v_i)$ non sélectionnée dans C_G, il existe une chaîne de u_i à v_i dans T (c.-à-d., une chaîne dans $\mathcal{C}_i$ qui ne traverse pas a_i). D'où :

Théorème 2.7. *Dans les cactus, le ratio d'intégrité pour MFEM (resp. pour MCM) est au plus* 4. *En outre, une solution pour MFEM (resp. pour MCM) réalisant ce rapport peut être calculée en temps polynomial.*

La famille d'instances donnée dans la figure 2.5 prouve que notre analyse de cet algorithme est fine, puisqu'il existe des instances où $\|C_G\|$ est égal à $4\|F_T\|$ (en prenant $d = 2$). Cependant, ceci n'implique pas nécessairement que les ratios d'intégrité pour MFEM et MCM sont fins. Notons que le théorème 2.7 est également vrai pour MFEMI. En outre, ce théorème prouve que le ratio d'intégrité pour MFEM est ramené à 4 quand le degré intérieur maximum est au plus 2, alors qu'il vaut $\Omega(\sqrt{n})$ quand le degré maximum est 3 [80, p. 17].

2.4 Bilan et problèmes ouverts

L'une des motivations de l'étude décrite dans ce chapitre était de comprendre les raisons qui rendaient possibles les résultats de Garg et al. dans les arbres. Une réponse intéressante a été apportée, puisque, d'après les résultats qui sont présentés ici, *tous* les résultats de Garg et al. restent valides dans les graphes de nombre cyclomatique borné (et nous avons vu que ce n'était pas vrai si l'on considérait d'autres généralisations classiques des arbres) : en particulier, MCDA reste polynomial (ce résultat unifie et généralise le cas des arbres et des anneaux, qui étaient les deux seules classes de graphes où ce problème était connu comme polynomial), et le ratio d'intégrité pour MFEM est borné par deux plus le double du nombre cyclomatique (c.-à-d., 2 dans les arbres). En outre, il est peu probable qu'il existe une classe de graphes plus générale que celle-ci où MCDA serait polynomial, étant donné qu'il est déjà APX-difficile dans les cactus triangulaires.

Par ailleurs, nous avons exhibé une classe de graphes planaires de largeur d'arbre bornée (mais quelconque) où le ratio d'intégrité de MCDA est borné (et même linéaire en la largeur d'arbre). Il faut remarquer que c'est le premier résultat de ce type généralisant le ratio constant de MCDA dans les arbres. Il faut également noter que ces graphes n'avaient jamais été définis ni étudiés à notre connaissance, malgré les nombreuses propriétés intéressantes qui leur sont associées, et il convient de se rappeler que, dans les graphes planaires à k niveaux (dont nos graphes sont une sous-classe naturelle), le meilleur ratio d'approximation (ainsi que la meilleure borne sur le ratio d'intégrité) connu est logarithmique en n [41].

Enfin, selon nous, l'un des intérêts majeurs des résultats d'approximation de ce chapitre est qu'ils reposent sur une étude paramétrée des graphes : autrement dit, ils ne font pas appel à des propriétés structurelles particulières qui ne sont vérifiées que par certaines classes de graphes (si l'on excepte la planarité, utile dans la deuxième partie du chapitre), mais à des paramètres caractéristiques, communs à tous les graphes (tous les graphes possèdent un nombre cyclomatique et tous les graphes planaires ont k niveaux d'arêtes pour un certain $k = O(n)$). Bien évidemment, nos résultats ne deviennent vraiment intéressants que si, malgré tout, ces paramètres sont petits. Il convient cependant de noter que les temps d'exécution de nos algorithmes approchés ne dépendent pas de ces paramètres (seuls les ratios d'approximation et d'intégrité en dépendent). En outre, nos algorithmes sont simples et utilisent comme sous-routines des algorithmes conçus pour les arbres : ainsi, toute modification de ces algorithmes (amélioration du temps de calcul, parallélisation, version en ligne, etc.) se répercute automatiquement sur les nôtres.

Nous terminons à présent ce chapitre en soulevant quelques questions ouvertes. L'une des plus intéressantes, qui découle directement de nos discussions précédentes, est la suivante : le ratio d'intégrité de MCDA dans les graphes planaires à k niveaux (ou même dans les graphes de largeur d'arbre borné) est-il constant ? Concernant la présente étude, une autre question capitale se pose : peut-on borner le ratio d'intégrité de MFEM (et non plus de MCDA) dans les graphes planaires à k niveaux d'arêtes ?

Chapitre 3

Multicoupes dans les graphes de largeur d'arbre bornée

3.1 Préliminaires

Dans ce chapitre, on s'intéresse à la complexité de MCM dans les graphes, orientés ou non, de largeur d'arbre bornée.

Nous commençons par quelques rappels. Pour $|\mathcal{L}| = 1$ (une seule liaison), MCM est équivalent au problème de la coupe minimum, et est donc polynomial. Il en est de même, dans les graphes non orientés, pour le problème CMTM avec deux terminaux (c'est-à-dire avec $|\mathcal{T}| = 2$). En outre, CMTM est APX-difficile pour $|\mathcal{T}| = 3$ dans les graphes non orientés [54], et pour $|\mathcal{T}| = 2$ dans les graphes orientés [78] : cela implique que MCM est APX-difficile pour $|\mathcal{L}| = 3$ dans les graphes non orientés, et pour $|\mathcal{L}| = 2$ dans les graphes orientés. Néanmoins, d'après [54], CMTM est polynomial dans les graphes de largeur d'arbre bornée (en utilisant des techniques classiques de programmation dynamique), alors que MCM reste APX-difficile dans les étoiles non pondérées (mais devient polynomial dans les arbres si $|\mathcal{L}|$ est borné) [80]. De surcroît, lorsque $|\mathcal{L}|$ est fixé, il existe une réduction polynomiale de MCM vers CMTM (malheureusement, elle ne conserve pas un certain nombre de propriétés, comme la planarité), que nous détaillons plus bas, car nous aurons à la réutiliser.

Dans la variante orientée de MCM, rappelons que l'objectif est d'enlever un ensemble d'arcs de poids minimum de façon à ce qu'aucun chemin orienté ne subsiste entre s_i et s'_i, pour chaque i. Ce problème est APX-difficile (la variante non orientée l'étant), mais Costa et al. ont montré qu'il devenait polynomial dans les arbres orientés (la matrice des contraintes étant totalement unimodulaire) [52]. Ils ont également montré que CMTM était polynomial dans les GSC.

Il faut noter qu'un des principaux résultats pour MCM est qu'il existe un PTAS pour ce problème dans les graphes non orientés qui sont non pon-

dérés et de largeur d'arbre et de degré maximum bornés (le problème étant seulement $\mathcal{NP}$-difficile dans ce cas, car il l'est dans les arbres binaires non pondérés ; nous détaillons cette astucieuse réduction plus bas), et que la suppression d'une des trois hypothèses (non pondéré, degré maximum borné, largeur d'arbre bornée) rend le problème APX-difficile [27]. Il a également été montré dans [27] que ce PTAS peut s'adapter à une variante de MCM dans les graphes orientés, définie dans [101] et que l'on appellera MCM-FC :

Définition 3.1. Problème MCM-FC : *l'objectif est d'enlever un ensemble d'arcs de poids minimum de façon à ce que, pour chaque i, aucun circuit ne contienne à la fois s_i et s'_i.*

Les principaux résultats d'approximation pour MCM sont qu'il existe un algorithme $O(\log|\mathcal{L}|)$-approché dans les graphes non orientés [79] (pour les arbres, un algorithme 2-approché est connu [80]) et un algorithme $O(\sqrt{n})$-approché dans les graphes orientés [88], alors que pour MCM-FC il existe un algorithme $O(\log^2|\mathcal{L}|)$-approché [101]. Rappelons que d'autres résultats d'approximation et de complexité concernant MCM et CMTM sont disponibles dans [26, 42, 52, 54, 91, 97, 130, 164].

Enfin, MCMS a été étudié dans [27], où il a été montré qu'il était polynomial dans les arbres non pondérés et qu'il admettait un PTAS dans les graphes non orientés et non pondérés de largeur d'arbre bornée ; en fait, le PTAS pour MCM est fondé sur ce résultat : les auteurs décrivent une réduction de MCM dans les graphes non orientés et non pondérés de largeur d'arbre et de degré maximum bornés vers MCMS dans les graphes non orientés et non pondérés de largeur d'arbre bornée, et montrent qu'elle préserve l'approximation. En outre, MCMS est $\mathcal{NP}$-difficile dans les graphes séries-parallèles (qui sont, rappelons-le, les graphes de largeur d'arbre 2) de degré maximum 3, et devient $\mathcal{NP}$-difficile dans les arbres non pondérés de degré maximum 4 *si on ne s'autorise pas à enlever des sommets terminaux* (nous qualifierons de *restreinte* cette variante).

Dans ce chapitre, nous montrons tout d'abord que, si $|\mathcal{L}|$ est fixé, MCM est polynomial dans les graphes non orientés de largeur d'arbre bornée (ce qui généralise le résultat de [80] pour le cas des arbres). Le deuxième résultat que nous montrons dans la section 3.2 est que MCMS est $\mathcal{NP}$-difficile dans les cactus triangulaires non pondérés ayant une largeur de chaîne et un degré maximum bornés (alors que, rappelons-le, il est polynomial dans les arbres non pondérés). Puis, la section 3.3 est consacrée au cas des graphes orientés. Le premier résultat de cette section est que MCM est $\mathcal{NP}$-difficile dans les GSC, et le deuxième est que MCM est APX-difficile dans les graphes non pondérés de largeur d'arbre orientée et de degré maximum bornés (contrairement à MCM-FC et au cas non orienté de MCM).

Le tableau 3.1 récapitule les résultats de complexité les plus importants pour MCM dans les graphes de largeur d'arbre bornée. Les résultats prouvés dans ce chapitre sont indiqués en gras et en italique.

Problème	Largeur d'arbre		Degré		Orientation		Pondération		$\|\mathcal{L}\|$		Résultat
	Bornée	?	Borné	?	Oui	Non	Oui	Non	Borné	?	
MCM	✓			✓		✓		✓		✓	APX-difficile même dans une étoile [80]
MCM		✓		✓		✓		✓	✓		APX-difficile [54]
CMTM	✓			✓		✓	✓			✓	Polynomial [54]
MCM	✓			✓		✓	✓		✓		***Polynomial*** (3.2.1)
MCMS	✓			✓		✓		✓		✓	PTAS [27]
MCMS	✓		✓			✓		✓		✓	$\mathcal{NP}$-difficile même si le graphe est série-parallèle [27]
MCMS	✓			✓		✓		✓		✓	Polynomial dans les arbres [27]
MCMS	✓		✓			✓		✓		✓	***$\mathcal{NP}$-difficile même dans les cactus de largeur de chaîne bornée*** (3.2.2)
MCM	✓			✓	✓		✓			✓	Polynomial dans les arbres (orientés) [52]
CMTM	✓			✓	✓		✓			✓	Polynomial dans les GSC [52]
MCM	✓		✓		✓			✓		✓	***$\mathcal{NP}$-difficile même dans les GSC dont le graphe non orienté sous-jacent est un cactus*** (3.3.1)
MCM	✓		✓			✓		✓		✓	PTAS & $\mathcal{NP}$-difficile (APX-difficile si degré, pondération ou largeur d'arbre quelconque) [27]
MCM-FC	✓		✓		✓			✓		✓	Idem
CMTMS	✓		✓		✓			✓		✓	***APX-difficile même dans les GSC***
MCM	✓		✓		✓			✓		✓	***APX-difficile*** (3.3.2)

TAB. 3.1 – Résumé des résultats de complexité et d'approximation pour MCM et ses variantes, relatifs à ce chapitre. Les "?" signifient que l'attribut concerné peut prendre une valeur quelconque, et les numéros des sections où les résultats sont établis sont indiqués entre parenthèses.

Résoudre MCM via CMTM, par énumération exhaustive des partitionnements. Dahlhaus et al. ont décrit dans [54] une méthode astucieuse pour réduire MCM à CMTM, lorsque le nombre de liaisons est fixé.

Il suffit pour cela de remarquer que, dans toute solution optimale de MCM, les $2|\mathcal{L}|$ terminaux sont répartis en q paquets (nous allons montrer que $q \leq \sqrt{2|\mathcal{L}|}+1$), tels que (i) pour chaque i, aucun paquet ne contient s_i et s'_i, et (ii) pour chaque paire de paquets, il existe un i tel que s_i est dans l'un des paquets et s'_i est dans l'autre (sinon on peut fusionner les deux paquets et obtenir un partitionnement qui reste valide). La propriété principale de ce partitionnement est que si l'on ajoute, pour chaque paquet, un nouveau sommet t_i (appelé *sommet additionnel*) relié par une nouvelle arête (appelée *arête additionnelle* et ayant une capacité suffisamment grande) à chaque terminal de ce paquet, la multicoupe minimum associée à ce partitionnement fournit une coupe multiterminale minimum sur l'instance de CMTM obtenue en prenant les sommets additionnels comme terminaux. Inversement, toute solution optimale pour cette instance de CMTM fournit une solution optimale pour l'instance initiale de MCM. Nous appellerons cette méthode (qui, remarquons-le, ne préserve pas nécessairement la planarité éventuelle de l'instance initiale) la *technique d'énumération des partitionnements* ; nous l'utiliserons en section 3.2.1.

Montrons à présent que $q \leq \sqrt{2|\mathcal{L}|}+1$. On utilise pour cela la propriété (ii) : elle implique que, si un partitionnement contient q paquets, c'est qu'il y a au moins $\frac{q(q-1)}{2}$ liaisons (une pour chaque paire de paquets). On a donc : $|\mathcal{L}| \geq \frac{q(q-1)}{2}$, soit $2|\mathcal{L}| \geq q(q-1) \geq (q-1)^2$. On en déduit le résultat souhaité. De cette façon, on montre le théorème suivant :

Théorème 3.1 ([54]). *Toute instance de MCM peut être résolue en résolvant au plus* $\frac{\left(\sqrt{2|\mathcal{L}|}+1\right)^{2|\mathcal{L}|}}{\left(\sqrt{2|\mathcal{L}|}+1\right)!}$ *instances de CMTM.*

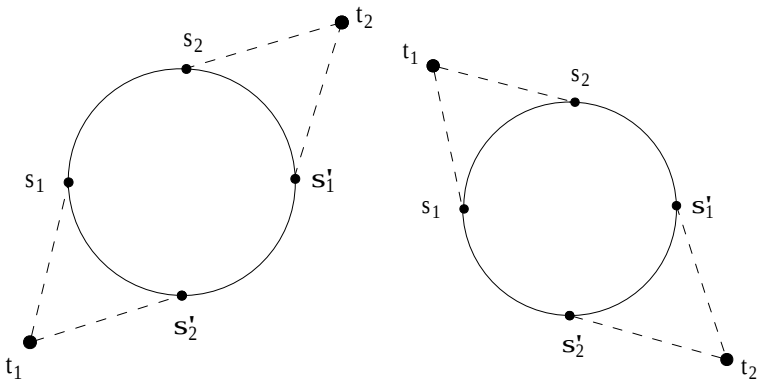
Malheureusement, cela ne signifie pas pour autant que l'on sache résoudre par un algorithme efficace les instances de CMTM obtenues (CMTM étant APX-difficile dans le cas général, même pour trois terminaux). Dahlhaus et al. ont utilisé ce résultat pour montrer que, lorsque $|\mathcal{L}|$ est fixé, on peut convertir un algorithme approché pour CMTM en un algorithme approché pour MCM. Nous l'utiliserons dans un cadre différent.

Pour finir, nous montrons comment appliquer cette technique dans les cas $|\mathcal{L}| = 2$ et $|\mathcal{L}| = 3$.

Lorsque l'on a seulement deux liaisons (s_1, s'_1) et (s_2, s'_2), l'énumération de tous les partitionnements possibles est immédiate : ils sont au nombre de deux, $\{\{s_1, s_2\}, \{s'_1, s'_2\}\}$ et $\{\{s_1, s'_2\}, \{s'_1, s_2\}\}$. En résolvant les deux instances de CMTM obtenues, on résout l'instance initiale de MCM : on peut remarquer que, dans ce cas ($|\mathcal{T}| = 2$), CMTM est équivalent au problème de la coupe minimum, et est donc polynomial. Ainsi, on a :

Théorème 3.2 ([163]). *Dans le cas non orienté, MCM peut se résoudre en temps polynomial lorsque $|\mathcal{L}| = 2$, en résolvant deux instances du problème de la coupe minimum.*

Une instance avec deux liaisons est donnée dans la figure 3.1.



(a) 1er partitionnement possible. (b) 2e partitionnement possible.

FIG. 3.1 – Une instance de MCM avec deux liaisons.

Quand $\mathcal{L} = \{(s_1, s'_1), (s_2, s'_2), (s_3, s'_3)\}$, il existe douze partitionnements :
$\{\{s_1, s_2\}, \{s_3, s'_1\}, \{s'_3, s'_2\}\}$; $\{\{s_1, s_2\}, \{s_3, s'_2\}, \{s'_3, s'_1\}\}$;
$\{\{s_1, s'_2\}, \{s_3, s_2\}, \{s'_3, s'_1\}\}$; $\{\{s_1, s'_2\}, \{s_3, s'_1\}, \{s'_3, s_2\}\}$;
$\{\{s_1, s_3\}, \{s_2, s'_1\}, \{s'_2, s'_3\}\}$; $\{\{s_1, s_3\}, \{s_2, s'_3\}, \{s'_2, s'_1\}\}$;
$\{\{s_1, s'_3\}, \{s_2, s'_1\}, \{s'_2, s_3\}\}$; $\{\{s_1, s'_3\}, \{s_2, s_3\}, \{s'_2, s'_1\}\}$;
$\{\{s_1, s_2, s_3\}, \{s'_1, s'_2, s'_3\}\}$; $\{\{s_1, s'_2, s_3\}, \{s'_1, s_2, s'_3\}\}$;
$\{\{s_1, s_2, s'_3\}, \{s'_1, s'_2, s_3\}\}$; $\{\{s_1, s'_2, s'_3\}, \{s'_1, s_2, s_3\}\}$.

Remarquons que, par exemple, nous ne considérons pas le partitionnement $\{\{s_1, s_2\}, \{s_3, s'_1\}, \{s'_3\}, \{s'_2\}\}$ (car les paquets $\{s'_3\}$ et $\{s'_2\}$ ne respectent pas la propriété (ii)) : il nous suffira de considérer à la place le partitionnement $\{\{s_1, s_2\}, \{s_3, s'_1\}, \{s'_3, s'_2\}\}$ (ou $\{\{s_1, s_2, s'_3\}, \{s'_1, s'_2, s_3\}\}$).

Preuve de complexité pour MCM dans les arbres binaires. Nous détaillons à présent la réduction donnée dans [27] pour montrer la $\mathcal{NP}$-difficulté de MCM dans les arbres binaires (c'est-à-dire de degré maximum 3) non pondérés. Cette réduction, très ingénieuse et pourtant très simple, se fait à partir du problème $\mathcal{NP}$-complet 3SAT [76] : étant données p variables booléennes $x_1, \ldots, x_p$ et q clauses $C_1, \ldots, C_q$ sur ces variables, chaque clause étant composée d'une disjonction de trois littéraux (un littéral étant une variable x_i ou son complément $\bar{x}_i$), le problème est de décider si on peut affecter une valeur à chaque variable (*vrai* ou *faux*) de façon à satisfaire toutes les clauses simultanément. Le principe de cette réduction est tellement puissant

qu'il peut facilement être adapté, comme nous le verrons ultérieurement, à d'autres cas que les arbres binaires.

La première étape est d'associer un "gadget" à chaque variable et un "gadget" à chaque clause, et de les relier ensuite de façon à obtenir un arbre binaire. Relier les "gadgets" entre eux ne pose pas de problème : c'est la définition des "gadgets" eux-mêmes qui est le cœur de cette réduction. L'exemple donné en figure 3.2 résume la structure des "gadgets" utilisés pour les variables et pour les clauses. Nous détaillons à présent quelques-unes de leurs propriétés :

- Le "gadget" associé à la i^e variable possède un sommet étiqueté x_i et un sommet étiqueté $\bar{x}_i$. Il existe une liaison entre deux sommets de ce "gadget" et deux façons de séparer les deux extrémités de cette liaison en enlevant une arête : dans le premier cas, x_i reste connecté au reste du graphe ; dans le deuxième cas, c'est $\bar{x}_i$ qui y reste connecté.
- Le "gadget" associé à la j^e clause possède, pour chacun des trois littéraux de la clause, un sommet étiqueté par ce littéral. Il existe deux liaisons reliant des sommets de ce "gadget" (les deux chaînes associées à ces liaisons dans l'arbre étant disjointes par les arêtes) et quatre façons de séparer les deux extrémités de chaque liaison en enlevant deux arêtes : pour chacune de ces façons, un sommet parmi les trois étiquetés par des littéraux reste connecté au reste du graphe (les deux premières façons étant associées à un littéral, la suivante à un autre, et la dernière au troisième).
- D'après les deux points précédents, il existe $p + 2q$ liaisons *internes*, c'est-à-dire entre des sommets appartenant à un même "gadget".
- Pour chaque i, il existe une liaison (dite liaison *externe*) entre le sommet étiqueté x_i (resp. étiqueté $\bar{x}_i$) dans le "gadget" de la i^e variable et tout autre sommet étiqueté x_i (resp. étiqueté $\bar{x}_i$).

Les liaisons internes permettent de garantir qu'au moins $p + 2q$ arêtes sont nécessaires dans toute multicoupe (puisque les chaînes associées à ces liaisons dans l'arbre sont disjointes par les arêtes) : il existe une multicoupe de taille $p + 2q$ ssi une arête est coupée dans le "gadget" de chaque variable et deux arêtes sont coupées dans le "gadget" de chaque clause. Les liaisons externes garantissent qu'il existe une multicoupe de taille $p + 2q$ ssi il existe une affectation de valeurs aux variables qui satisfait toutes les clauses de l'instance de 3SAT correspondante.

En effet, s'il existe une telle affectation, alors, pour chaque i, on coupe, dans le "gadget" associé à x_i, l'arête adjacente au sommet étiqueté x_i (resp. étiqueté $\bar{x}_i$) si x_i est *vrai* (resp. *faux*) dans l'affectation. En outre, pour chaque j, dans le "gadget" de la clause C_j, on coupe (en coupant deux arêtes) les deux liaisons internes de façon à ce que le terminal qui reste connecté au reste du graphe soit étiqueté par un littéral *vrai* dans l'affectation (il y en a nécessairement un). On obtient alors une multicoupe de valeur $p + 2q$.

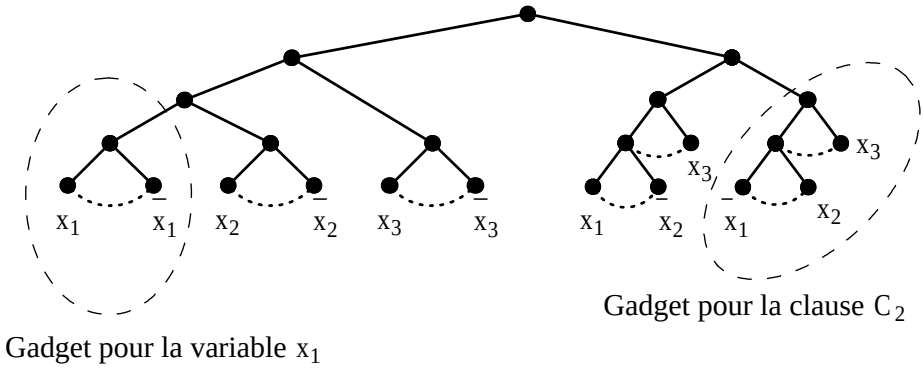


FIG. 3.2 – Réduction de 3SAT à MCM dans les arbres binaires (les liaisons internes sont en pointillés) pour l'instance $C_1 \wedge C_2$, avec $C_1 = x_1 \vee \bar{x}_2 \vee x_3$ et $C_2 = \bar{x}_1 \vee x_2 \vee x_3$.

Réciproquement, s'il existe une multicoupe de valeur $p + 2q$, on obtient une affectation valide en posant, pour chaque i, $x_i = vrai$ si l'arête adjacente au sommet étiqueté x_i dans la clause de la variable x_i est coupée, $x_i = faux$ sinon (rappelons que seule une des deux arêtes peut être coupée). D'où :

Théorème 3.3 ([27]). *MCM est $\mathcal{NP}$-difficile dans les arbres binaires non pondérés.*

3.2 Graphes non orientés

3.2.1 Polynomialité de MCM avec un nombre fixé de liaisons

Dans cette section, nous montrons que MCM est polynomial dans les graphes de largeur d'arbre bornée, si le nombre de liaisons est borné. Rappelons que si le graphe n'est pas de largeur d'arbre bornée ou si le nombre de liaisons n'est pas fixé, le problème devient APX-difficile [54, 80].

Théorème 3.4. *Si $|\mathcal{L}|$ est fixé, MCM est polynomial dans les graphes de largeur d'arbre bornée.*

Démonstration. On utilise la technique d'énumération des partitionnements détaillée dans l'introduction, ainsi que le fait que, d'après [54], CMTM peut être résolu en temps polynomial dans les graphes de largeur d'arbre bornée, à l'aide de techniques classiques de programmation dynamique. Avant l'introduction des sommets et arêtes additionnels, le graphe (que l'on supposera connexe sans perte de généralité) est de largeur d'arbre bornée par hypothèse, mais il nous reste à prouver que c'est toujours le cas une fois qu'ils ont été ajoutés. Nous utilisons pour cela le lemme suivant, dont la preuve est simple :

Lemme 3.1. *Soit $G = (S, A)$ un graphe connexe non orienté et soit G' le graphe obtenu à partir de G en ajoutant un nouveau sommet $\tilde{v}$ et des arêtes entre $\tilde{v}$ et certains sommets du graphe G. Alors $la(G') \leq la(G) + 1$.*

Démonstration. Étant donnée une décomposition arborescente Ψ de G, on construit une décomposition arborescente Ψ' pour G' en ajoutant à chaque sac de Ψ le sommet $\tilde{v}$. Il n'est pas difficile de se rendre compte que Ψ' est effectivement une décomposition arborescente pour G' (car, comme $\tilde{v}$ est dans chaque sac, toute arête $(u, \tilde{v})$ est dans un sac et le sous-graphe de l'arbre de décomposition associé à $\tilde{v}$ est l'arbre lui-même) et que la largeur de Ψ' est égale à la largeur de Ψ plus un. Ceci conclut la preuve. □

Ce lemme implique que la largeur d'arbre du graphe obtenu à partir de G en introduisant les sommets et arêtes additionnels est au plus la largeur d'arbre de G plus le nombre de sommets additionnels, q. Puisque $q \leq \sqrt{2|\mathcal{L}|} + 1$ et que $|\mathcal{L}|$ est fixé, ceci conclut la preuve du théorème 3.4. □

Précisons que nous nous sommes aperçu plusieurs mois après avoir trouvé ce résultat (courant 2005) qu'il avait été montré indépendamment dans [87], sans parvenir à établir qui l'avait obtenu en premier. Cependant, bien que les résultats des auteurs de ce papier soient plus complets que les nôtres, puisqu'ils montrent également que MCMS est polynomial dans ce cas, notre approche est plus simple et plus rapide. En effet, ils réduisent MCM à un problème plus général que CMTM, ce qui a deux conséquences : la première est qu'ils énumèrent de l'ordre de $2|\mathcal{L}|^{2|\mathcal{L}|}$ partitionnements, car ils doivent examiner tous les partitionnements contenant au plus $2|\mathcal{L}|$ paquets, alors que nous nous contentons d'en énumérer $O\left(\frac{\left(\sqrt{2|\mathcal{L}|}+1\right)^{2|\mathcal{L}|}}{\left(\sqrt{2|\mathcal{L}|}+1\right)!}\right)$; la seconde est qu'ils doivent concevoir un algorithme pour résoudre ce problème plus général que CMTM (alors que nous nous contentons de réutiliser les résultats de Dahlhaus et al. [54], ce qui raccourcit considérablement la preuve). En contrepartie, ils montrent la polynomialité de ce problème plus général dans les graphes de largeur d'arbre bornée, alors que notre méthode ne permet pas d'établir ce résultat.

3.2.2 Complexité de MCMS

Călinescu et al. montrent dans [27] que MCMS est *polynomial dans les arbres* et $\mathcal{NP}$-difficile (mais pas APX-difficile) dans les graphes non pondérés de largeur d'arbre et de degré maximum bornés et, pour la variante restreinte, dans les arbres non pondérés de degré maximum et de largeur de chaîne bornés. Comme MCMS admet un PTAS dans les graphes non pondérés de largeur d'arbre et de degré maximum bornés, on pourrait se demander si l'ajout de restrictions supplémentaires ne permettrait pas d'obtenir des cas

polynomiaux plus généraux que les arbres. Nous montrons dans cette section qu'une telle chose est très improbable, puisqu'une adaptation simple de la preuve de complexité donnée dans [27] pour MCM dans les arbres binaires non pondérés permet de montrer que MCMS reste $\mathcal{NP}$-difficile dans les graphes non pondérés de degré maximum et de largeur de chaîne bornés, même si le graphe est un cactus triangulaire et si chaque sommet n'apparaît qu'un nombre constant de fois dans la décomposition arborescente en chaîne (cette notion a été introduite dans [57] et définie comme la propriété de *Bounded Persistence Pathwidth*).

On adapte la réduction utilisée dans [27, Théorème 11, pp. 351-353] (et décrite dans la figure 3.2) en ajoutant une arête entre les extrémités de chaque liaison interne. Alors, dans toute solution, une des deux extrémités sera enlevée et le reste de la preuve de validité est identique.

Notons que la famille d'instances utilisée dans la preuve est bien une famille de cactus triangulaires non pondérés de degré maximum borné (car celle utilisée dans la preuve de [27, Théorème 11] est une famille d'arbres non pondérés de degré maximum borné). En outre, pour cette famille, on peut construire une décomposition arborescente comme suit : on inclut le "gadget" associé à chaque variable dans un sac et le "gadget" associé à chaque clause dans un sac (voir figure 3.3). L'arbre de la décomposition est une chaîne $B_1, \ldots, B_j, \ldots$ (la largeur de chaîne est donc bornée par $6 - 1 = 5$), où les B_i sont les sacs, et chaque sommet y apparaît au plus 3 fois, ce qui conclut la preuve. On a donc le théorème suivant :

Théorème 3.5. *MCMS est $\mathcal{NP}$-difficile dans les cactus triangulaires non pondérés de largeur de chaîne et de degré maximum bornés.*

Enfin, remarquons qu'une modification très simple (voir la figure 3.3) permet de s'assurer que le degré maximum est 3 (la largeur de chaîne étant alors bornée par 6).

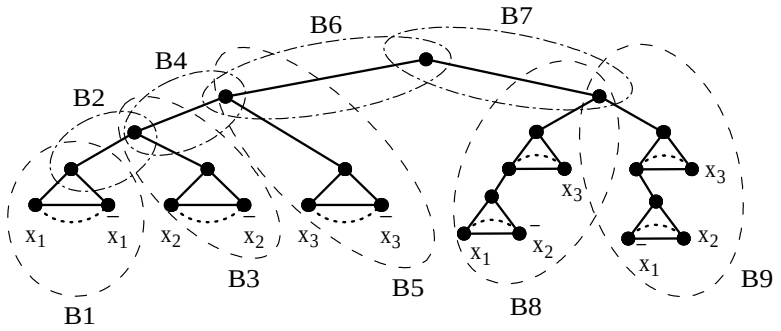


FIG. 3.3 – Réduction de 3SAT à MCMS (les liaisons internes sont en pointillés) pour l'instance $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$.

3.3 Graphes orientés

Dans cette section, nous étudions la complexité de MCM dans les GSC et dans les graphes orientés de largeur d'arbre et de degré maximum bornés, et, en particulier, nous répondons à la question de savoir si les idées données dans [27] pour MCM-FC pourraient s'appliquer à MCM ou non. Il est important de noter qu'aucune réduction polynomiale d'un problème à l'autre n'est connue ; donc, aucun n'est théoriquement plus "difficile" que l'autre. Néanmoins, nous montrons une réduction simple pour le cas où $|\mathcal{L}|$ est fixé :

Théorème 3.6. *Lorsque le nombre de liaisons $|\mathcal{L}|$ est fixé, MCM-FC se réduit polynomialement à MCM dans les graphes orientés.*

Démonstration. L'idée est que, dans toute solution optimale de MCM-FC, étant donnée une liaison (s_i, s'_i), soit il n'y a plus de chemins de s_i à s'_i, soit il n'y a plus de chemins de s'_i à s_i. Par conséquent, on peut, pour chaque liaison, essayer les deux possibilités ; cela signifie que l'on a à résoudre $2^{|\mathcal{L}|}$ (soit $O(1)$ si $|\mathcal{L}|$ est fixé) instances de MCM pour résoudre l'instance initiale de MCM-FC. □

On pourrait se demander s'il n'existe pas également une réduction de MCM vers MCM-FC. En effet, supposons qu'on se donne une instance de MCM, et qu'on relie ensuite s'_i à s_i par un arc de très grand poids, pour tout i. Si on résout l'instance de MCM-FC obtenue, on va chercher, pour tout i, à supprimer tous les chemins de s_i à s'_i, car supprimer tous ceux de s'_i à s_i aurait un coût trop important (à cause de l'arc de très grand poids que l'on a ajouté). Résoudre cette instance de MCM-FC équivaut donc à résoudre l'instance de MCM obtenue en ajoutant les arcs de très grand poids. Cependant, il est important de noter que les arcs ajoutés créent des chemins qui n'existaient pas dans l'instance initiale de MCM : par conséquent, les deux instances de MCM (l'instance initiale, et celle que l'on résoudrait effectivement en résolvant l'instance de MCM-FC obtenue) ne sont pas équivalentes. La figure 3.4 illustre ce fait. En conclusion, la transformation que nous venons de décrire ne constitue *pas* une réduction valide de MCM vers MCM-FC.

3.3.1 Complexité de MCM dans les GSC

Rappelons que MCM dans les graphes orientés a été étudié dans [26, 88, 113], ainsi que dans [90], où les auteurs conjecturent que MCM n'est pas significativement plus simple dans les GSC (rappelons que les GSC sont, en particulier, les graphes de largeur d'arbre orientée égale à 0). Dans [27], il est montré que MCM-FC est $\mathcal{NP}$-difficile dans les graphes orientés non pondérés de largeur d'arbre et de degré maximum bornés (notons que MCM-FC est trivial dans les GSC : la valeur optimale est toujours égale à 0).

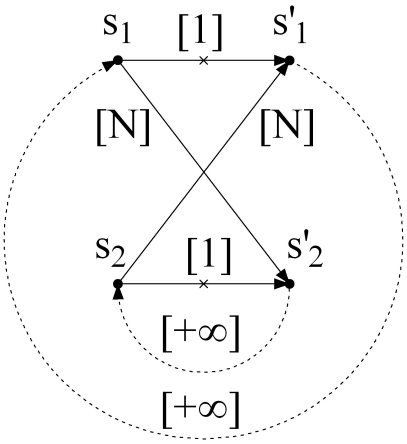


FIG. 3.4 – Transformation (incorrecte) d'une instance de MCM en une instance de MCM-FC. Avant ajout des arcs de très grand poids (en pointillés), la valeur optimale de l'instance de MCM est 2 (les deux arcs de poids 1). Après ajout, elle vaut $N + 2$ (les deux arcs de poids 1 et un arc de poids N).

Dans cette section, nous montrons que MCM reste $\mathcal{NP}$-difficile dans les GSC non pondérés, planaires et ayant un degré maximum borné (c.-à-d., même lorsque la largeur d'arbre orientée est 0), en adaptant de nouveau la réduction utilisée dans [27] pour montrer que MCM est $\mathcal{NP}$-difficile dans les arbres binaires (non orientés). En outre, nous verrons que le résultat reste vrai même si le graphe non orienté sous-jacent est un cactus. Il est intéressant de noter que, en comparaison, CMTM est polynomial dans les GSC et MCM est polynomial dans les arbres orientés (c'est-à-dire si le graphe non orienté sous-jacent est un arbre) [52]. Nous montrerons dans la section suivante que MCM reste APX-difficile dans les graphes orientés non pondérés de largeur d'arbre et de degré maximum bornés.

Nous suivons le schéma de preuve donné dans en section 3.1, et initialement dans [27]. La réduction se fait donc à partir de 3SAT, et il y a un "gadget" pour chaque variable et un "gadget" pour chaque clause. On note $x_1, \ldots, x_p$ les variables, et $C_1, \ldots, C_q$ les clauses. En outre, comme dans [27], pour chaque i, il existe une liaison *externe* entre le sommet étiqueté x_i (resp. étiqueté $\bar{x}_i$) dans le "gadget" de la i^e variable et tout autre sommet étiqueté x_i (resp. étiqueté $\bar{x}_i$). Le point essentiel de la preuve est de construire ces "gadgets" (car ici on a besoin de "gadgets" orientés et non plus non orientés).

La figure 3.5 montre la construction orientée correspondant à la construction non orientée donnée dans [27, figure 4(c)] (et en figure 3.2). La caractéristique remarquable de cette construction orientée est qu'elle possède les mêmes propriétés de base que la construction non orientée originale. Cependant, notre preuve est légèrement différente.

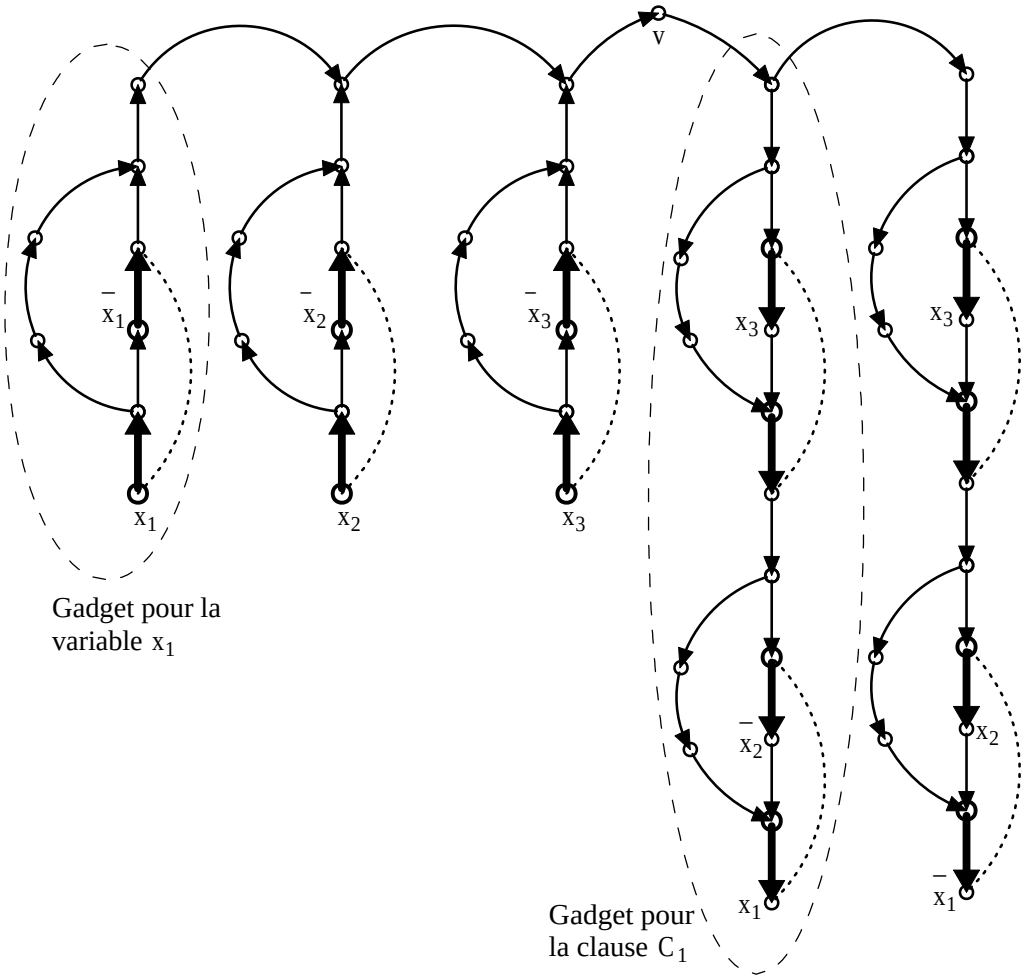


FIG. 3.5 – Réduction de 3SAT à MCM (les liaisons internes sont en pointillés) pour l'instance $C_1 \wedge C_2$, avec $C_1 = x_1 \vee \bar{x}_2 \vee x_3$ et $C_2 = \bar{x}_1 \vee x_2 \vee x_3$.

Comme dans la preuve originale, les liaisons internes garantissent qu'au moins $p + 2q$ arêtes sont requises dans toute multicoupe, et les liaisons externes garantissent qu'il existe une multicoupe de taille exactement $p + 2q$ (rappelons que, dans notre problème, séparer une source de son puits nécessite qu'il ne subsiste plus aucun chemin orienté de la source vers le puits) ssi il existe une affectation B de valeurs aux variables qui satisfait toutes les clauses de l'instance de 3SAT correspondante. En effet, si B existe, alors on construit une multicoupe de taille $p + 2q$ comme suit :

- Pour chaque i, on coupe, dans le "gadget" associé à la variable x_i, l'arc en gras adjacent au sommet étiqueté x_i si x_i est *vrai* dans B, et l'arc en gras adjacent au sommet étiqueté $\bar{x}_i$ sinon ;
- Pour chaque clause, il y a quatre façons de couper deux des quatre arcs en gras de façon à couper les deux liaisons internes, et, pour chacune de ces façons, il n'y a qu'un seul sommet parmi les trois sommets étiquetés par des littéraux que l'on peut atteindre à partir de v, c.-à-d., vers lequel il subsiste un chemin issu de v ; on coupe de telle façon que le sommet que l'on puisse atteindre soit un littéral *vrai* dans B.

Réciproquement, s'il existe une multicoupe de valeur $p+2q$, alors on met à *vrai* les littéraux x_i tels que, dans le "gadget" de la i^e variable, c'est l'arc en gras adjacent au sommet x_i qui est coupé ; on met les autres littéraux à *faux* (même ceux pour lesquels aucun des deux arcs en gras n'est coupé ; c'est une convention). On obtient ainsi B. En effet, étant donné un "gadget" associé à une clause, quelle que soit la façon dont on coupe deux de ses arcs pour que les deux liaisons internes soient coupées, il reste au moins un des trois sommets étiquetés par des littéraux qui peut être atteint par un chemin issu de v. Ainsi, si ce littéral est x_j (resp. $\bar{x}_j$) pour un certain j, alors l'arc en gras adjacent à x_j (resp. $\bar{x}_j$) dans le "gadget" de la j^e variable doit être coupé ; par conséquent x_j est *vrai* (resp. *faux*) et la clause est satisfaite. D'où :

Théorème 3.7. *MCM est $\mathcal{NP}$-difficile dans les GSC non pondérés.*

Notons que ce théorème reste vrai si le graphe non orienté sous-jacent est un cactus biparti de degré maximum 3.

En outre, si le degré maximum est 2, alors le graphe non orienté sous-jacent est l'union disjointe de cycles et de chaînes ; par conséquent, MCM est polynomial dans ce cas (même si le graphe orienté n'est pas un GSC), puisqu'il est polynomial dans les chemins et les anneaux [52].

Par ailleurs, si le graphe est non pondéré, si $|\mathcal{L}|$ est fixé et si le degré maximum est borné (par un entier d), alors MCM est polynomial. En effet, une solution admissible est obtenue en sélectionnant les arcs adjacents aux terminaux (soit au plus $d|\mathcal{L}|$ arcs) : par conséquent, toute solution optimale est de valeur (et donc de cardinalité) au plus $d|\mathcal{L}|$, et il suffit donc d'énumérer toutes les solutions contenant au plus $d|\mathcal{L}|$ arcs et de garder la meilleure.

On peut également remarquer que, si le graphe non orienté sous-jacent est un graphe biparti (les ensembles de la bipartition étant S_1 et S_2) et si tous les arcs sont orientés de S_1 vers S_2 (c'est-à-dire si pour tout arc $(u, v) \in A$ on a $u \in S_1$ et $v \in S_2$), alors MCM est trivialement polynomial. En effet, dans ce cas, il y a au plus un arc entre s_i et s'_i pour tout i.

En réalité, ce cas est un cas particulier des graphes orientés **étagés** (*layered digraphs* en anglais [90]) : un graphe orienté étagé est un graphe $G = (S, A)$ dont les sommets peuvent être partitionnés en $q \geq 2$ ensembles $S_1, \ldots, S_q$ tels que, pour tout arc $(u, v) \in A$, on a $u \in S_i$ et $v \in S_{i+1}$ pour un $i \in \{1, \ldots, q-1\}$ (les S_j peuvent être vus comme des étages empilés les uns sur les autres ; tout arc relie alors un des étages à l'étage suivant). Ces graphes sont sans circuits, et, pour $q = 2$, on retrouve le cas précédent. En fait, résoudre MCM (ou MFEM) dans les graphes orientés étagés équivaut à le résoudre dans les GSC. En effet, étant donné un GSC, on commence par déterminer une numérotation (ou ordre) des sommets telle que pour tout arc (v_i, v_j) on a $i < j$ (*ordre topologique*) ; ensuite, on remplace tout arc (v_i, v_j) tel que $j > i + 1$ par un chemin de longueur $j - i$ (dont tous les arcs sont pondérés par le même poids que l'arc (v_i, v_j)).

Une simple adaptation de la preuve du théorème 3.7 montre que MCM est $\mathcal{NP}$-difficile dans les graphes orientés étagés, même si $q = 16$ (en accrochant tous les "gadgets" au même sommet v). Une autre adaptation de cette preuve permet de montrer que, si tous les étages ont une taille bornée par un entier p et si p est fixé et q quelconque, alors MCM est aussi $\mathcal{NP}$-difficile (en fait, même si $p = 11$) ; remarquons que ce cas généralise les chemins. (Si p *et* q sont fixés, la taille du graphe est fixée, et le cas est donc sans intérêt.) Pourtant, si p *et* $|\mathcal{L}|$ sont fixés, un algorithme très simple basé sur la programmation dynamique [14] permet de résoudre MCM en temps polynomial. On note $n_i = |S_i|$, et on définit

$$V(i, x_{1,1}, \ldots, x_{|\mathcal{L}|,1}, x_{1,2}, \ldots, x_{|\mathcal{L}|,2}, \ldots, x_{1,n_i}, \ldots, x_{|\mathcal{L}|,n_i})$$

comme étant la valeur d'une multicoupe minimum, qui, pour i donné et pour tout j, l, sépare s_j du l^e sommet de S_i si et seulement si $x_{j,l} = 0$. (Pour i donné, le nombre de valeurs différentes pour $V(i, \cdot)$ est $O(2^{p|\mathcal{L}|})$.) Il n'est pas difficile de voir que le calcul des $V(i, \cdot)$ pour un $i > 1$ ne nécessite que de connaître les $V(i-1, \cdot)$. En fait, on obtient les $V(i, \cdot)$ en énumérant, pour chaque $V(i-1, \cdot)$, toutes les combinaisons d'arcs se trouvant entre S_{i-1} et S_i : il y a au plus p sommets dans S_{i-1} et p sommets dans S_i, donc au plus p^2 arcs entre S_{i-1} et S_i ; par conséquent, il y a $O(2^{p^2})$ combinaisons d'arcs à considérer pour chaque vecteur $(x_{1,1}, \ldots, x_{|\mathcal{L}|,1}, x_{1,2}, \ldots, x_{|\mathcal{L}|,2}, \ldots, x_{1,n_i}, \ldots, x_{|\mathcal{L}|,n_i})$.

On obtient donc en tout $O(2^{p(p+|\mathcal{L}|)})$ valeurs : pour chacun des vecteurs $(x_{1,1}, \ldots, x_{|\mathcal{L}|,1}, x_{1,2}, \ldots, x_{|\mathcal{L}|,2}, \ldots, x_{1,n_i}, \ldots, x_{|\mathcal{L}|,n_i})$, on garde alors la solution de meilleure valeur (et on réduit donc le nombre de valeurs à $O(2^{p|\mathcal{L}|})$). (Remarquons que, s'il existe j tel que $s'_j \in S_i$, on ne doit garder que les vecteurs pour lesquels s_j est effectivement séparé de s'_j.) Puis, on recommence :

on calcule les $V(i+1,\cdot)$ à partir des $V(i,\cdot)$. Lorsque l'on atteint le dernier étage, on garde la meilleure solution qui sépare s_j de s'_j pour tout j : on obtient ainsi une solution optimale. En outre, les deux paramètres p et $|\mathcal{L}|$ étant fixés, cet algorithme est polynomial (et même FPT vis-à-vis de ces deux paramètres).

3.3.2 APX-difficulté de MCM

Les résultats de cette section répondent par la négative à la question suivante : peut-on réutiliser, pour résoudre MCM, les idées du PTAS pour MCM-FC dans les graphes orientés non pondérés de largeur d'arbre et de degré maximum bornés ? En effet, ce PTAS ne peut pas être adapté à MCM, puisque, dans la preuve du fait que l'algorithme de programmation dynamique fournit bien une solution admissible pour MCM-FC, l'un des arguments clés (voir [27, p. 348]) utilise le fait que tout circuit commence et finit dans le même ensemble W_r, car il commence et finit au même sommet. Il est clair que cet argument ne peut pas être utilisé dans notre cas, puisque nous considérons des chemins (les deux extrémités d'un chemin peuvent appartenir à deux ensembles quelconques W_r et W_s) et non des circuits.

En réalité, nous montrons que MCM est APX-difficile dans les graphes orientés non pondérés de largeur d'arbre et de degré maximum bornés (alors que le cas non orienté admet un PTAS), ce qui implique que MCM-FC est plus facile que MCM même dans ce cas particulier (puisqu'il existe un PTAS). Pour démontrer ceci, nous donnons une réduction préservant l'approximation à partir du problème CouvertureParLesSommets dans les graphes de degré maximum borné, qui est connu pour être APX-difficile [135]. Cette réduction a été en partie inspirée par celle donnée dans [78] pour réduire CouvertureParLesSommets à CMTMS dans les graphes non orientés, mais on peut remarquer que la réduction n'aurait pas pu être adaptée pour CMTM dans les graphes orientés non pondérés de largeur d'arbre et de degré maximum bornés.

Soit un graphe non orienté G ayant n sommets et un degré maximum borné. Rappelons qu'une couverture de G est un ensemble de sommets S' de G tel que pour chaque arête $a = (u, v)$ de G, on a $u \in S'$ ou $v \in S'$. On oriente G comme suit : on numérote de façon arbitraire les n sommets $v_1, \ldots, v_n$, et on transforme ensuite chaque arête (v_i, v_j) en un arc (v_i, v_j) (si $i < j$) ou en un arc (v_j, v_i) (si $j < i$). De cette façon, $v_1, \ldots, v_n$ définit un *ordre topologique* sur les sommets (c'est-à-dire que $i < j$ pour chaque arc (v_i, v_j)), et le graphe orienté D que l'on obtient est un GSC (et a donc une largeur d'arbre orientée égale à 0). Si on définit une couverture dans D comme un ensemble de sommets dans D qui couvre tous les arcs de D, les couvertures de G coïncident avec celles de D. Cela montre d'ores et déjà que CMTMS est APX-difficile dans les GSC (alors que CMTM est polynomial dans ce cas [52]), même si le degré maximum est borné, car les couvertures

dans D sont les coupes multiterminales sur les sommets (si chaque sommet est un terminal).

À présent, nous transformons D en une instance orientée de MCM. On ajoute n nouveaux sommets : $t_1, t_2, \ldots, t_n$, et, pour chaque i, on relie v_i et t_i par le "gadget" à cinq arcs donné en figure 1.4 (pour un i, on ajoute deux nouveaux sommets a_i et b_i, et les cinq arcs sont (v_i, a_i), (b_i, v_i), (a_i, b_i), (t_i, a_i) et (b_i, t_i).) De cette façon, on obtient un nouveau graphe G' (voir figure 3.6 pour un exemple). Alors, il est facile de voir que l'on a une couverture de taille N dans D (et donc dans G) si et seulement si on a une multicoupe de taille N dans G', où les liaisons sont (t_i, t_j) pour chaque arc (v_i, v_j) de D. En effet, étant donnée une couverture S' de taille N pour G, on peut construire une multicoupe de taille N dans G' comme suit : pour chaque i, on sélectionne (a_i, b_i) dans la coupe si et seulement si $v_i \in S'$. Inversement, étant donnée une multicoupe de taille N pour G', soit il n'y a que des arcs du type (a_i, b_i) dans la coupe, soit on peut transformer cette coupe en une nouvelle coupe de taille au plus N et ne contenant que des arcs de ce type : si un arc (v_i, v_j) est coupé, alors on le remplace par (a_i, b_i) ; si un arc du "gadget" reliant v_i et t_i est coupé, alors on le remplace par (a_i, b_i). On peut maintenant définir une couverture de taille N dans G en incluant tous les v_i tels que (a_i, b_i) appartient à la multicoupe pour G'. La description de la réduction préservant l'approximation est à présent terminée.

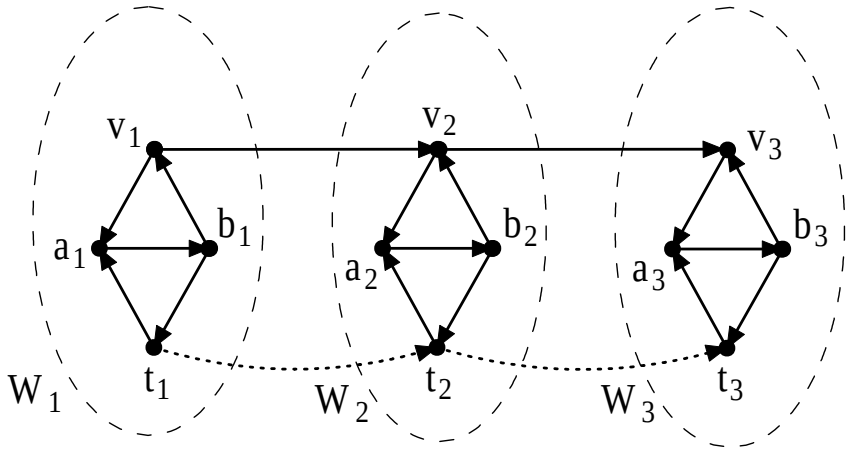


FIG. 3.6 – Transformation d'un graphe G (ici, une chaîne avec 3 sommets v_1, v_2, v_3) en G'. Les deux liaisons sont les arcs en pointillés.

Il est facile de voir que G' est non pondéré et a un degré maximum borné (puisque le degré maximum de G est borné). Il nous reste à montrer que G' a une largeur d'arbre orientée bornée. On construit une décomposition arborescente orientée Ψ pour G' de la façon suivante. L'arborescence de la décomposition est un chemin avec n sommets. L'ensemble W_r associé au

r^e sommet de ce chemin est $W_r = \{v_r, a_r, b_r, t_r\}$ (W_1 étant la racine de l'arborescence) et les ensembles associés aux arcs de ce chemin sont vides. Alors, les ensembles W_r définissent bien une partition des sommets de G'. Supposons que tous les sommets $v_1, v_2, \ldots, v_n$ de D se trouvent sur un axe horizontal dans cet ordre : alors, d'après la façon dont D a été construit, tout arc reliant v_i à v_j pour $i < j$ sera orienté de gauche à droite. Ainsi, il est facile de voir que, pour chaque r, tout chemin qui va d'un sommet dans W_{r_1} avec $r_1 \geq r$ à un sommet dans W_{r_2} avec $r_2 \geq r_1$ ne visite aucun sommet d'un des ensembles situés à gauche de r (c.-à-d., un ensemble $W_{r'}$ avec $r' < r$). Donc, Ψ est effectivement une décomposition arborescente pour G'. En outre, sa largeur est $4 - 1 = 3$. Par conséquent :

Théorème 3.8. *MCM est APX-difficile dans les graphes orientés non pondérés de largeur d'arbre et de degré maximum bornés.*

3.4 Bilan et problèmes ouverts

Nous avons montré dans ce chapitre un ensemble de résultats concernant la complexité (et l'approximabilité) de MCM dans les graphes de largeur d'arbre bornée. En particulier, nous avons montré que MCM est polynomial dans les graphes de largeur d'arbre bornée, si le nombre de liaisons est fixé (ce qui généralise le résultat de Garg et al. pour les arbres [80]). En outre, nous avons montré que MCM est $\mathcal{NP}$-difficile dans les GSC, même si le graphe non orienté sous-jacent est un cactus (alors que, si c'est un arbre, le problème est polynomial). Enfin, nous avons montré que MCM est APX-difficile dans les graphes non pondérés de largeur d'arbre orientée et de degré maximum bornés, alors que le cas non orienté admet un PTAS : cela fournit un exemple de plus où ces deux cas ont des comportements divergents. Notons que ce cas est d'ailleurs d'un intérêt tout particulier puisqu'il correspond à des hypothèses très restrictives, et qu'il met en lumière les différences entre les versions orientées et non orientées de la largeur d'arbre.

Un problème ouvert intéressant serait de déterminer la complexité de MCM lorsque le graphe est un GSC et le nombre de liaisons fixé.

Chapitre 4

Maximiser le multiflot entier dans les anneaux

4.1 Préliminaires

Dans ce chapitre, nous étudions le problème MFEM dans les anneaux (qui sont les arbres d'anneaux 2-sommet-connexes). Rappelons que ce problème est polynomial dans les étoiles et dans les chaînes (où la matrice des contraintes de sa formulation en PL est totalement unimodulaire), mais qu'il devient $\mathcal{NP}$-difficile dans les arbres [80] et les arbres d'anneaux.

Les réseaux en anneau ont beaucoup d'applications dans le domaine des télécommunications (voir, par exemple, [6, 20, 47, 74, 116, 150]), et certains problèmes de routage dans des anneaux sont connus pour être $\mathcal{NP}$-difficiles ([47, 150]). Le cas d'un anneau à capacité uniforme a également été étudié pour MFED [116]. Cependant, à notre connaissance, la complexité de MFEM dans les anneaux restait jusqu'à présent inconnue. Notons que quelques propriétés utiles ainsi qu'un algorithme linéaire résolvant MFEM dans les anneaux à capacité uniforme sont donnés dans [20].

En outre, il semble que la seule étude concernant le polytope associé à MFEM soit due à Brunetta et al. [25]. Ils donnent une famille d'inégalités valides, qu'ils appellent en anglais *multi-handle comb inequalities*, et montrent que certaines d'entre elles définissent des facettes.

Dans ce chapitre, nous commençons par montrer que MFEM est polynomial dans les anneaux en étendant une approche utilisée par Sai Anand et Erlebach [6] pour résoudre des cas particuliers du *problème d'admission et de routage d'appels* dans un anneau (section 4.2). Nous introduisons ensuite de nouvelles inégalités valides pour MFEM dans les anneaux, et donnons des exemples montrant qu'elles ne sont pas suffisantes pour caractériser entièrement le polytope associé (P), même dans des anneaux uniformes (section 4.3). Tout d'abord, pour conclure ces préliminaires, nous rappelons quelques propriétés importantes des anneaux.

Définitions et propriétés. Comme nous l'avons déjà évoqué quelques lignes auparavant, quelques propriétés utiles des anneaux sont données dans [20]. D'abord, un anneau non orienté peut être transformé en un anneau orienté en doublant le nombre de liaisons (propriété (α)) : il suffit de choisir une orientation arbitraire (dans le sens des aiguilles d'une montre ou en sens inverse) et de remplacer chaque liaison initiale (s_i, s'_i) par deux nouvelles liaisons $(s_{1,i}, s'_{1,i})$ et $(s_{2,i}, s'_{2,i})$, telles que $s_{1,i}$ et $s'_{2,i}$ se trouvent en s_i et $s'_{1,i}$ et $s_{2,i}$ se trouvent en s'_i. Donc, on supposera dans la suite que tous nos anneaux sont orientés, et, étant donnée une liaison (s_i, s'_i), on désignera par p_i l'unique chemin reliant s_i à s'_i. En outre, nous dirons qu'une liaison (s_i, s'_i) **croise** une liaison (s_j, s'_j) si p_i et p_j ont au moins un arc en commun.

Ensuite, si le chemin p_i d'une liaison (s_i, s'_i) est inclus dans le chemin p_j d'une liaison (s_j, s'_j), on peut supprimer (s_j, s'_j) et acheminer de s_i à s'_i toute unité de flot qui transitait auparavant entre s_j et s'_j (propriété (β)). Cela n'affecte pas la valeur optimale de l'instance considérée.

Enfin, dans les anneaux uniformes (c'est-à-dire, où tous les arcs ont la même capacité), s'il existe un sommet contenant au plus un terminal (une source ou un puits), alors l'un des deux arcs adjacents est inutile, et l'anneau peut être réduit (propriété (γ)).

Les propriétés (α), (β) et (γ) impliquent que, dans les anneaux uniformes, on peut supposer sans perte de généralité que l'on a :
- (i) sur chaque sommet se trouvent une source et un puits ;
- (ii) tous les chemins dans $\bigcup_{i\in\{1,\ldots,|\mathcal{L}|\}} p_i$ ont la même longueur ;
- (iii) le nombre de liaisons, $|\mathcal{L}|$, est égal au nombre de sommets.

Dans tout ce chapitre, pour formuler MFEM comme un PL, nous utiliserons le modèle (IPL-MFEM-2) décrit en section 1.3.3 (qui, dans ce cas particulier, a un nombre linéaire de contraintes et de variables). Remarquons aussi que, d'après (α), on peut effectivement supposer sans perte de généralité que l'anneau est orienté : f_i représente donc la quantité de flot acheminée par le chemin p_i (en supposant les p_i numérotés dans le sens des aiguilles d'une montre). On notera m le nombre d'arcs de l'anneau, a_j le j^e arc et c_j sa capacité. En outre, on désignera par (PL-MFEM-2) la relaxation continue de (IPL-MFEM-2).

Une remarque utile pour la suite de ce chapitre est que les réductions associées aux propriétés (α), (β) et (γ) conservent le caractère basique d'une solution optimale.

4.2 Résolution de MFEM

Nous montrons à présent que MFEM est polynomial dans les anneaux en utilisant des idées similaires à celles proposées dans [6, pp. 26–29] pour

résoudre des cas particuliers du *problème d'admission d'appels dans les anneaux* : en substance, ce problème est équivalent (dans le cas où les appels sont non pondérés) à MFEM dans les anneaux orientés lorsque le flot routé de s_i à s'_i peut être 0 ou 1 pour chaque i. Nous prouvons le théorème suivant :

Théorème 4.1. *MFEM est polynomial dans les anneaux.*

Démonstration. On effectue une recherche dichotomique sur la valeur de $\sum_{i=1}^{|\mathcal{L}|} f_i$ ($\leq |\mathcal{L}|c_{max}$, où c_{max} est le maximum des capacités des arcs de l'anneau). Une fois que la valeur de cette somme est fixée et égale à un entier F, on peut ajouter au PL la contrainte $(C0)$: $\sum_{i=1}^{|\mathcal{L}|} f_i = F$. Notre problème devient alors un problème de décision : existe-t-il une solution admissible ? En outre, si l'on considère la matrice $0-1$ des contraintes du PL initial, il existe deux sortes de lignes : soit les 1 sont consécutifs (et alors les 0 ne le sont pas), soit les 0 sont consécutifs (et alors les 1 ne le sont pas) ; c'est la *propriété des 1 circulaires*. En effet, la j^e ligne de la matrice correspond à la contrainte de capacité sur le j^e arc a_j : $\sum_{i \ / \ p_i \text{ traverse } a_j} f_i \leq c_j$. Comme les f_i sont numérotés dans le sens des aiguilles d'une montre, soit les numéros de ceux qui traversent c_j sont consécutifs (et donc la j^e ligne de la matrice est constituée de 0 et d'un bloc de 1 consécutifs), soit f_1 et $f_{|\mathcal{L}|}$ sont parmi ces f_i (comme, par exemple, dans la somme $f_{|\mathcal{L}|-1} + f_{|\mathcal{L}|} + f_1 + f_2$) et les numéros des f_i traversant c_j sont donc consécutifs jusqu'au numéro $|\mathcal{L}|$ puis à partir du numéro 1 (et donc la j^e ligne de la matrice est constituée de 1 et d'un bloc de 0 consécutifs). Pour chaque contrainte C impliquant des liaisons non consécutives (et donc où les 1 ne sont pas consécutifs), on définit une nouvelle contrainte $(C') \leftarrow (C0) - (C)$ et on supprime (C) : dans (C'), tous les 1 sont consécutifs. Par conséquent, on obtient un problème équivalent dont la matrice $0-1$ des contraintes vérifie la *propriété des 1 consécutifs*, et qui est donc totalement unimodulaire. Ainsi, comme le membre de gauche de ce système (le vecteur des capacités) est entier, il existe une solution entière si et seulement si il existe une solution réelle, et il est bien connu que l'existence d'une telle solution peut être décidée en temps polynomial [5]. □

En outre, on peut remarquer que notre approche implique que :

Corollaire 4.1. *Dans les anneaux, l'écart entre la valeur du multiflot continu maximum et la valeur du multiflot entier maximum est strictement inférieur à 1.*

Démonstration. L'existence d'une solution fractionnaire de valeur F^* (où F^* est la valeur optimale en continu) implique l'existence d'une solution fractionnaire de valeur F pour chaque réel $F \leq F^*$ et, en particulier, pour $\lfloor F^* \rfloor$. Mais, d'après la preuve du théorème précédent, cela implique l'existence d'une solution entière de valeur $\lfloor F^* \rfloor$. □

Enfin, il convient de noter que notre résultat est en fait plus général que son énoncé pourrait le laisser entendre. Notre approche permet en fait de résoudre tout PL dont tous les coefficients de la fonction objectif sont égaux, et dont la matrice des contraintes vérifie la propriété des 1 circulaires.

4.3 Une famille d'inégalités valides pour MFEM

Soit G un anneau orienté, soit $L \geq 2$ un entier, et soit $l_0 = (s_0, s'_0), l_1 = (s_1, s'_1), \ldots, l_{h-1} = (s_{h-1}, s'_{h-1})$ un ensemble de h liaisons telles que, pour chaque i, l_i croise à la fois les $L-1$ liaisons $l_{(i-L+1) \bmod h}, \ldots, l_{(i-1) \bmod h}$ et les $L-1$ liaisons $l_{(i+1) \bmod h}, \ldots, l_{(i+L-1) \bmod h}$. Pour chaque i, on note $c_{min}(i)$ le minimum des capacités des arcs situés entre s_i et $s'_{(i-L+1) \bmod h}$ et f_i le flot routé entre s_i et s'_i. Alors, pour chaque $i \in \{0, \ldots, h-1\}$, on a :

$$f_{(i-L+1) \bmod h} + \cdots + f_{(i-1) \bmod h} + f_i \leq c_{min}(i)$$

Si on fait la somme sur i, chaque flot f_i est compté L fois, donc on a :

$$L \sum_{i=0}^{h-1} f_i \leq \sum_{i=0}^{h-1} c_{min}(i)$$

Comme les flots sont entiers, l'inégalité suivante est valide pour MFEM :

$$\sum_{i=0}^{h-1} f_i \leq \left\lfloor \frac{1}{L} \sum_{i=0}^{h-1} c_{min}(i) \right\rfloor$$

Si on renumérote les flots et les capacités de 1 à h pour simplifier les notations, on obtient :

$$\sum_{i=1}^{h} f_i \leq \left\lfloor \frac{1}{L} \sum_{i=1}^{h} c_{min}(i) \right\rfloor \tag{4.1}$$

Nous donnons à présent deux exemples montrant que, même dans les anneaux uniformes, ces inégalités ne sont pas suffisantes pour caractériser complètement le polytope de MFEM. Rappelons que l'on peut supposer que tout anneau uniforme vérifie (i), (ii) et (iii). Donc, supposons d'abord que l'on ajoute à la formulation (PL-MFEM-2) uniquement l'inégalité valide appartenant à la famille donnée dans (4.1) et qui vérifie $h = |\mathcal{L}| = m$, c'est-à-dire celle qui implique toutes les liaisons. Si c désigne la capacité de chaque arc de l'anneau, cette inégalité peut s'écrire ainsi :

$$\sum_{i=1}^{m} f_i \leq \left\lfloor \frac{mc}{L} \right\rfloor \tag{4.2}$$

Désignons par (PL-MFEM-2A) le PL obtenu à partir de (PL-MFEM-2) en ajoutant (4.2). Rappelons qu'une solution de base d'un PL avec $|\mathcal{L}|$

variables est une solution qui sature au moins $|\mathcal{L}|$ contraintes linéairement indépendantes (sous la condition que le polytope associé est de pleine dimension). En outre, il n'est pas difficile de s'apercevoir que le polytope associé à (PL-MFEM-2A) est de pleine dimension (c'est-à-dire de dimension $|\mathcal{L}|$).

Supposons à présent que $c = L = 3$ et $m = |\mathcal{L}| = 5$, par exemple. La solution

$$f_1 = f_3 = f_4 = \frac{3}{2}, f_2 = f_5 = 0$$

est une solution de base admissible (puisqu'elle sature les contraintes $f_2 \geq 0$, $f_5 \geq 0$, $f_4 + f_5 + f_1 \leq 3$, $f_1 + f_2 + f_3 \leq 3$ et $f_2 + f_3 + f_4 \leq 3$, et la matrice associée est inversible), et c'est une solution fractionnaire. Si l'on suppose ensuite que, pour la même instance, on ajoute *toutes* les inégalités valides (4.1) (remarquons à ce propos que l'inégalité valide $f_1+f_3+f_4 \leq 4$ est violée par la solution précédente), alors, on peut se demander si, au moins dans les anneaux uniformes, une telle famille serait suffisante pour caractériser complètement (P), le polytope de MFEM. Malheureusement, comme le montre l'exemple suivant, il subsiste des solutions de base fractionnaires même dans ce cas très particulier.

Considérons la solution suivante pour l'instance précédente :

$$f_1 = f_3 = f_4 = \frac{4}{3}, f_2 = f_5 = \frac{1}{3}$$

Montrons que c'est une solution de base admissible : elle sature les contraintes $f_4+f_5+f_1 \leq 3$, $f_1+f_2+f_3 \leq 3$, $f_2+f_3+f_4 \leq 3$, $f_3+f_4+f_5 \leq 3$ et l'inégalité valide $f_1 + f_3 + f_4 \leq \lfloor \frac{3\times 3}{2} \rfloor$, et il n'est pas difficile de vérifier que, d'une part, la matrice associée est inversible, et que, d'autre part, c'est une solution admissible. En outre, c'est une solution fractionnaire.

En dépit de ces résultats négatifs, l'équation (4.2) est suffisante pour garantir que toute solution de base optimale est entière dans les anneaux uniformes. En effet, d'après (i), (ii) et (iii) (voir la section 4.1), dans un anneau avec une capacité uniforme c, la solution fractionnaire admissible $f_i = \frac{c}{L}$ pour chaque i est optimale et a une valeur égale à $\frac{mc}{L}$. Par conséquent, il existe également au moins une solution fractionnaire de valeur $\lfloor \frac{mc}{L} \rfloor$, et cela implique que la valeur optimale de (PL-MFEM-2A) est $\lfloor \frac{mc}{L} \rfloor$ (on a donc $\sum_{i=1}^{|\mathcal{L}|} f_i = \lfloor \frac{mc}{L} \rfloor$). Ainsi que nous l'avons expliqué dans la section précédente, on peut alors transformer ce PL en un PL équivalent ayant une matrice de contraintes totalement unimodulaire (et donc, dont toutes les solutions de base sont entières) ; en outre, toute solution de base pour l'un est solution de base pour l'autre. En d'autres termes, toute solution de base de (PL-MFEM-2A) est également entière dans ce cas. Cela montre que, dans les anneaux uniformes, l'ajout de l'inégalité valide (4.2) à (PL-MFEM-2) est suffisant pour garantir que toute solution de base est entière.

4.4 Conclusion

Dans ce chapitre, nous avons étudié MFEM dans les anneaux. Ce problème est polynomial dans les chaînes (et, dans ce cas, toute solution de base est entière car la matrice des contraintes est TU), et nous avons montré qu'il reste polynomial dans les anneaux. Cependant, le polyèdre associé (P) semble beaucoup plus difficile à étudier dans ce cas : nous avons donné une nouvelle famille d'inégalités valides pour MFEM dans les anneaux (alors que, à notre connaissance, il n'existait auparavant qu'une seule famille d'inégalités valides connue pour (P) [25]), et fourni des exemples montrant qu'elles ne suffisent pas à caractériser complètement (P), même dans les anneaux uniformes. Néanmoins, il convient de remarquer que, pour arriver à une telle caractérisation, il serait peut-être nécessaire de considérer d'autres inégalités valides, notamment des inégalités plus générales associées aux polyèdres représentés par des matrices vérifiant la propriété des 1 circulaires. Une étude plus approfondie prenant en considération de telles inégalités valides serait donc une perspective intéressante.

Un autre problème ouvert intéressant serait d'étudier de façon plus approfondie le polytope (P), et notamment dans des topologies généralisant les anneaux.

Deuxième partie

Graphes planaires

Chapitre 5

Étude des problèmes dans les grilles

5.1 Introduction et état de l'art

Dans ce chapitre, nous traitons de problèmes de multiflot entier et de multicoupe dans des grilles. Il convient de remarquer qu'il existe dans la littérature de nombreux articles concernant des problèmes (de décision) de chemin ou de multiflot entier dans les graphes planaires et en particulier dans les grilles (on ne considère ici que des grilles rectangulaires), à cause de leurs nombreuses applications dans la conception de circuits VLSI (voir, par exemple, [112, pp. 47–100], [118], [129, pp. 625–712], et l'ensemble des références citées dans la section *Grid graphs* du livre *Combinatorial Optimization* de Schrijver [151, pp. 1323–1325]).

En outre, ces articles traitent souvent du cas avec capacités unitaires, c'est-à-dire du problème CDA. Dans le cas où les s_i et les s'_i sont sur le bord, Frank donne des conditions nécessaires et suffisantes (vérifiables en temps polynomial) pour l'existence de $|\mathcal{L}|$ chemins disjoints par les arêtes, ainsi qu'un algorithme polynomial pour construire les chemins lorsqu'ils existent ([69] et [112, pp. 47-100]), mais, si les terminaux peuvent se situer n'importe où dans la grille, le problème devient $\mathcal{NP}$-complet [114]. Marx a même montré qu'il restait $\mathcal{NP}$-complet dans les grilles eulériennes [125].

Essentiellement deux types de problèmes d'optimisation peuvent être associés au problème de décision CDA.

D'abord, on peut chercher à déterminer quel est le nombre maximum de paires (s_i, s'_i) que l'on peut router via des chemins disjoints par les arêtes (problème MLCDA). Dans les grilles, d'après les résultats de complexité donnés précédemment, ce que l'on peut espérer de mieux en temps polynomial est résoudre des cas particuliers ou concevoir des algorithmes approchés. Concernant les cas particuliers, Chan et Chin donnent dans [30] un algorithme efficace pour trouver le nombre maximum de chemins disjoints par

les arêtes dans des grilles où tout sommet issu d'un ensemble de sources peut être couplé avec tout sommet issu d'un ensemble de puits. Concernant les algorithmes approchés, Kleinberg et Tardos donnent dans [102] un algorithme approché avec un ratio constant pour MLCDA dans une classe de graphes *quasi-eulériens* (qu'ils nomment, en anglais, *densely embedded* et *nearly-eulerian*), qui généralisent les grilles. Aggarwal et al. étudient l'approximation du problème de router des permutations de sommets d'une grille via des chemins disjoints par les sommets [4].

Ensuite, on peut chercher à déterminer un routage de longueur totale minimum. Plus formellement, étant donné un graphe $G = (S, A)$ avec des capacités entières sur les arêtes et $|\mathcal{L}|$ paires (source s_i, puits s'_i) de sommets terminaux, on définit le problème suivant [48] :

Définition 5.1. Problème de multichemins à longueur minimum (noté **MCLM**) : *l'objectif est de relier chaque paire* (s_i, s'_i) *par un chemin tout en respectant les contraintes de capacité, de façon à minimiser la somme des longueurs des* $|\mathcal{L}|$ *chemins (où la* **longueur** *d'un chemin est le* nombre *d'arêtes dont il est composé).*

Une autre variante de ce problème est de trouver un routage minimisant la longueur du plus long chemin. Les deux variantes sont $\mathcal{NP}$-difficiles dans les graphes planaires, même si les terminaux sont sur la face extérieure, si le degré maximum est 4 et si tout sommet qui n'est pas sur la face extérieure a un degré pondéré pair [24], alors que CDA est polynomial dans ce cas, même si le degré maximum n'est pas borné [70] (ce résultat est une extension d'un célèbre résultat d'Okamura et Seymour [134]). Néanmoins, un cas polynomial intéressant de MCLM a été étudié par Formann et al. [66] : ils supposent que le graphe est une grille particulière qu'ils appellent *canal dense* (voir la définition formelle en section 5.3), et que toutes les arêtes ont une capacité de 1. Dans ce cas, le résultat de Frank [69] évoqué précédemment implique que le problème n'admet aucune solution admissible, sauf si toutes les liaisons peuvent être routées verticalement (cas trivial) ou s'il y a une ligne verticale supplémentaire sur l'un des côtés (gauche ou droite) de la grille. Donc, pour le cas non trivial, les auteurs considèrent une grille à laquelle on a ajouté une ligne verticale sans terminal sur l'un des côtés et montrent que MCLM est alors polynomial.

Dans la suite, après avoir donné en section 5.2 quelques définitions et résultats préliminaires, nous résolvons MCLM dans le cas d'un canal dense avec des capacités horizontales et verticales quelconques (section 5.3) et un cas particulier de MCDA relatif au cas étudié par Frank dans [69] (section 5.5). Des résultats de complexité pour MCM et MFEMI sont donnés en section 5.4.

5.2 Définitions, notions et résultats préliminaires

Dans cette section, nous donnons les définitions et les résultats fondamentaux utilisés dans toute la suite de ce chapitre.

5.2.1 Définitions

Dans ce qui suit, nous considérons un graphe qui est une grille (rectangulaire) non orientée avec m lignes horizontales (ou simplement *lignes*) et n lignes verticales (ou *colonnes*). La première ligne est la plus haute et la première colonne est la plus à gauche (voir [66, 69] pour une définition formelle d'une grille rectangulaire).

La frontière de la grille est composée de quatre *bords* : la première et la m^e ligne horizontale (les **bords horizontaux**) et la première et la n^e ligne verticale (les **bords verticaux**). Nous aurons à considérer des grilles **bilatérales**, dans lesquelles les terminaux sont deux à deux distincts et chaque terminal est situé sur un des deux bords horizontaux, c'est-à-dire, soit sur la première ligne (la plus haute, dite ligne *supérieure*), soit sur la dernière (la plus basse, dite ligne *inférieure*). Deux terminaux situés sur le même bord horizontal sont dits **du même bord**. Un sommet de la ligne supérieure (resp. inférieure) est dit **sommet supérieur** (resp. **sommet inférieur**).

Étant donné un terminal z, nous dénoterons par $col(z)$ sa colonne. Étant donnée une liaison $l_i = (s_i, s'_i)$, on pose $source(l_i) = col(s_i)$, et on dira que l_i est **vers la gauche** si $col(s_i) > col(s'_i)$, **vers la droite** si $col(s_i) < col(s'_i)$, et **verticale** autrement. Par exemple, dans la figure 5.1, la liaison (s_3, s'_3) est vers la gauche et la liaison (s_5, s'_5) est vers la droite.

Étant donnée une liaison non verticale (s_i, s'_i), s_i (resp. s'_i) est **le terminal de gauche** si $col(s_i) < col(s'_i)$ (resp. $col(s_i) > col(s'_i)$), **le terminal de droite** sinon. On dit qu'une liaison l est **strictement plus D-longue** qu'une liaison l' si et seulement si le terminal de droite de l est situé à droite du terminal de droite de l' (on dira seulement **plus D-longue** si on autorise les terminaux de droite à se trouver sur la même colonne). On définit une liaison (**strictement**) **plus G-longue** de façon similaire, en remplaçant *droite* par *gauche* : par exemple, dans la figure 5.4, la liaison (s_{10}, s'_{10}) est plus D-longue et strictement plus G-longue que la liaison (s_9, s'_9).

Une grille bilatérale est **pleine** si tous les sommets des lignes supérieure et inférieure sont des sommets terminaux ; dans ce cas, $|\mathcal{L}| = n$ (voir les exemples donnés dans les figures 5.4 et 5.5).

Définition 5.2. *Une* **bande verticale** *(resp. une* **bande horizontale***) est la région (et les arêtes) se trouvant entre deux lignes verticales (resp. deux lignes horizontales) consécutives.*

On notera v_j (resp. h_j) la j^e bande verticale (resp. horizontale), la première étant la plus à gauche (resp. la plus haute). Notons que v_j est entre les j^e et $j+1^e$ colonnes.

Définition 5.3. *La* **densité** [66] d_j *d'une bande verticale* v_j *est le nombre de liaisons qui la "traversent" :*

$$d_j = |\{(s_i, s'_i) \text{ t.q. } col(s_i) \leq j < col(s'_i) \text{ ou } col(s'_i) \leq j < col(s_i)\}|$$

Une bande verticale v_j est **saturée** si $d_j = m$. La **densité de la grille** est $d = \max_{j \in \{1,\ldots,n-1\}} \{d_j\}$ et est au plus $|\mathcal{L}|$ (voir les figures 5.1 et 5.4).

Définition 5.4. *Une* **région dense** *est un ensemble, maximal au sens de l'inclusion, de bandes verticales consécutives de densité maximale.*

Une région dense sera notée $[\alpha, \beta]$, où α (resp. β) est la colonne la plus à gauche (resp. la colonne le plus à droite) adjacente à cette région : $d_j = d$ pour $j \in \{\alpha, \ldots, \beta - 1\}$ et

$$d_{\alpha-1} = d_\beta = d - 2 = d_\alpha - 2 = d_{\beta-1} - 2 \tag{5.1}$$

Dans la figure 5.1, on a $d = 4$ et une région dense $[\alpha, \beta] = [2, 4]$.

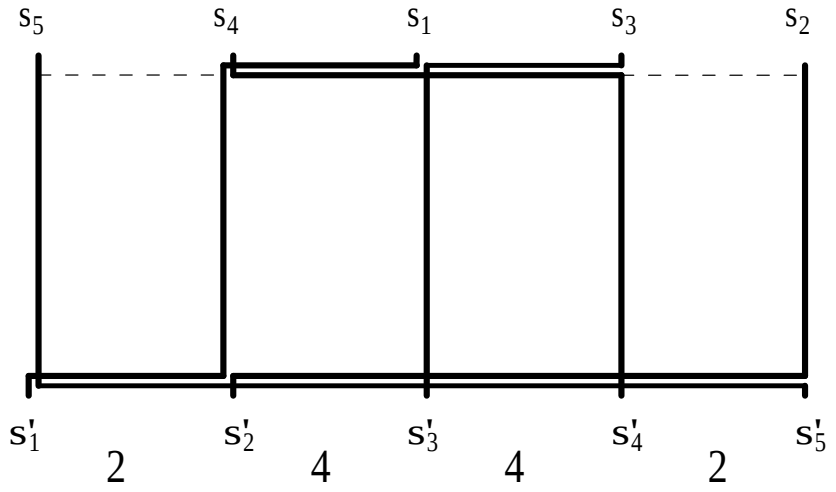


FIG. 5.1 – Un routage dans une grille avec 2 lignes et 5 liaisons.

5.2.2 Résultats préliminaires

Le premier résultat important dont nous aurons besoin concerne les densités d'une grille bilatérale. La preuve a été donnée dans [165], mais la propriété suivante, aisément vérifiable, en résume l'essentiel :

$$\forall j \in \{0, \ldots, n-2\},\ d_{j+1} = d_j + n^G_{j+1} - n^D_{j+1} \tag{5.2}$$

où $d_0 = 0$ et $n_j^G \in \{0, 1, 2\}$ (resp. $n_j^D \in \{0, 1, 2\}$) est le nombre de liaisons dont le terminal de gauche (resp. le terminal de droite) est sur la j^e colonne (notons que, pour tout j, $n_j^G + n_j^D + 2n_j^V \leq 2$ où $n_j^V \in \{0, 1\}$ est le nombre de liaisons verticales sur la j^e colonne). On a donc le lemme suivant :

Lemme 5.1. *Dans une grille bilatérale pleine, étant donnée une bande verticale v_j de densité d_j, d_{j+1} est égale à :*
- $d_j + 2$ *si et seulement si* $n_j^G = 2$ *;*
- $d_j - 2$ *si et seulement si* $n_j^D = 2$ *;*
- d_j *si et seulement si* $n_j^G = n_j^D = 1$ *ou* $n_j^V = 1$.

Ceci implique, en particulier :

Proposition 5.1. *Une grille pleine a la propriété que d_j est paire pour toute bande verticale v_j.*

Nous présentons à présent un résultat de Frank sur les routages dans des grilles bilatérales.

Théorème 5.1 (Frank)**.** *Soit une grille rectangulaire bilatérale non pondérée ayant m lignes et une densité d. Supposons qu'il y ait au moins une liaison non verticale. Alors, toutes les liaisons peuvent être reliées par des chemins disjoints par les arêtes si et seulement si la grille vérifie :*

soit	$m > d$	**et**		*il y a au moins un sommet non terminal sur un bord horizontal*	
soit	$m \geq d$				(5.3)
		et	**soit**	*il existe une liaison unilatérale*	(5.4)
			soit	*il existe un sommet extrême*	(5.5)
			soit	*il existe deux sommets non séparés*	(5.6)

où *une liaison unilatérale* est une liaison dont les deux terminaux sont du même bord, *un sommet extrême* est un sommet non terminal situé sur un bord horizontal qui est soit à gauche de la bande verticale saturée la plus à gauche, soit à droite de la bande verticale saturée la plus à droite ; enfin *deux sommets non séparés* sont deux sommets non terminaux du même bord, qui ne sont pas séparés par une bande verticale saturée.

Preuve (schéma). En fait, Frank a montré dans [112, pp. 47-100] que, dans une grille bilatérale, une condition suffisante est de vérifier (5.3) et (5.4). La condition (5.3) est nécessaire, car c'est la condition de coupe appliquée aux bandes verticales. En outre, il est prouvé dans [69] que, dans les grilles **strictement bilatérales** (c'est-à-dire, dans les grilles bilatérales où (5.4) n'est pas vérifiée), la seconde partie du théorème 5.1 devient *si et seulement si la grille vérifie* (5.3) *et soit* (5.5) *soit* (5.6). □

Enfin, nous aurons besoin d'un célèbre résultat d'Okamura et Seymour. D'abord, nous rappelons la définition du problème bien connu de *multiflot avec demandes* (voir aussi en section 1.3).

Définition 5.5. *Supposons que l'on se donne un* graphe support $G = (S, A)$ *et un* graphe de demandes $H = (\mathcal{T}, \mathcal{L})$*, dont les sommets* $\mathcal{T} \subseteq S$ *sont les terminaux et dont les arêtes sont les liaisons. Chaque arête* $a \in A$ *est pondérée par une capacité* $U(a)$ *et chaque arête (ou liaison)* l *de* $\mathcal{L}$ *est pondérée par une demande* $D(l)$*. Le* problème du multiflot avec demandes *consiste à décider s'il est possible de router toutes les demandes tout en respectant les contraintes de capacité.*

Si on requiert des flots entiers, le problème est MFED ; sinon, c'est MFCD (voir la section 1.3). Par convention d'écriture, nous utiliserons $\mathcal{L}$ au lieu de $H = (\mathcal{T}, \mathcal{L})$ ($\mathcal{T}$ étant implicite), et l'expression *ensemble de demandes* au lieu de *graphe de demandes*. Pour $L \subseteq A$ et $L' \subseteq \mathcal{L}$, on définit $U(L) = \sum_{l \in L} U(l)$ et $D(L') = \sum_{l \in L'} D(l)$. Alors, rappelons qu'une condition nécessaire de solubilité bien connue est la *condition de coupe* :

$$\forall X \subseteq V,\ U(\delta_G(X)) \geq D(\delta_{\mathcal{L}}(X))$$

Un sommet v est **impair** si $U(\delta_G(\{v\})) + D(\delta_{\mathcal{L}}(\{v\}))$ (c.-à-d., son degré pondéré dans $G + H$) est impair. En outre, on dira qu'une grille G munie d'un ensemble de demandes $\mathcal{L}$ ($(G, \mathcal{L})$ en abrégé), ainsi que d'une fonction de capacité U et d'une fonction de demande D, vérifie la **condition eulérienne** ssi il n'a aucun sommet impair, c'est-à-dire ssi

$$\forall v \in V,\ U(\delta_G(\{v\})) + D(\delta_{\mathcal{L}}(\{v\})) \text{ est pair} \tag{5.7}$$

Okamura et Seymour ont prouvé le théorème suivant [134] :

Théorème 5.2 (Okamura et Seymour)**.** *Soit* $G = (S, A)$ *un graphe support planaire avec un ensemble de demandes* $\mathcal{L}$*. Soient* U *et* D *les fonctions de capacité et de demande respectivement. Supposons que les sommets terminaux sont sur la face extérieure de* G*. Alors, il existe un multiflot continu admissible si et seulement si la condition de coupe est vérifiée. En outre, si* U *et* D *sont à valeurs entières et si* $(G, \mathcal{L}, U, D)$ *vérifie la condition eulérienne, alors il existe un multiflot entier.*

Dans toute la suite, nous supposerons qu'une instance $\mathcal{I}$ du problème de multiflot avec demandes est donnée par $\mathcal{I} = (G, \mathcal{L}, U, D)$ avec G, $\mathcal{L}$, U et D définis comme ci-dessus.

5.3 Résolution de MCLM dans des canaux denses pondérés

5.3.1 Préliminaires

Dans cette section, nous donnons un algorithme polynomial très simple pour résoudre le problème MCLM (défini en section 5.1) dans une famille de canaux denses (voir la définition formelle plus bas). Ces résultats font suite à l'étude menée dans la thèse de Maria Zrikem [165] : les résultats de la section 5.3.2 y étaient déjà présents, mais nous en donnons de nouvelles preuves (plus concises) ; les résultats des sections 5.3.3 et 5.3.4 constituent, quant à eux, des extensions nouvelles de ces résultats initiaux. L'ensemble a fait l'objet d'une publication commune (avec Marie-Christine Costa et Christophe Picouleau) [18].

Définissons à présent formellement un *canal dense pondéré* :

Définition 5.6. *Un* **canal dense pondéré** *est une grille pleine strictement bilatérale (les sources étant sur le bord supérieur et les puits sur le bord inférieur ; voir figure 5.1) où toute arête horizontale (resp. verticale) de la grille est pondérée par le même entier positif U_h (resp. U_v).*

Un canal dense pondéré a donc $|\mathcal{L}|$ liaisons et $|\mathcal{L}|$ colonnes. En outre, rappelons que Formann et al. ont étudié MCLM dans un canal dense pondéré où $U_h = U_v = 1$ [66]. Nous supposerons systématiquement et sans perte de généralité que, dans un canal dense, les liaisons sont numérotées de telle façon que $col(s'_i) = i$, pour chaque $i \in \{1, \ldots, |\mathcal{L}|\}$.

Dans la suite de la section 5.3, nous donnons un algorithme polynomial glouton pour résoudre MCLM dans les canaux denses lorsque U_v et U_h sont quelconques (à l'exception du cas $U_h = U_v = 1$, traité dans [66]). La principale caractéristique de notre algorithme est qu'il route chaque liaison via un plus court chemin (et donc, contrairement au cas $U_h = U_v = 1$, aucune ligne verticale supplémentaire n'est requise). Ainsi, de façon évidente, la longueur totale est minimum.

On peut remarquer qu'une condition nécessaire pour qu'il existe une solution admissible pour une instance donnée est la condition de coupe appliquée aux bandes verticales, qui peut s'écrire :

$$\textit{pour tout } j,\ mU_h \geq d_j, \text{ c.-à-d., } m \geq \lceil \tfrac{d}{U_h} \rceil \tag{5.8}$$

On commencera par traiter le cas où U_h est pair et U_v vaut 1 (section 5.3.2), puis on résoudra les cas où U_h est impair et U_v vaut 1 (section 5.3.3) et où $U_v \geq 2$ (section 5.3.4).

5.3.2 Résolution du cas où U_h est pair et $U_v = 1$

Nous allons commencer par traiter le cas où $U_h = 2$, et nous verrons ensuite comment généraliser l'approche au cas où U_h est pair.

Pour $U_h = 2$, (5.8) devient $m \geq \frac{d}{2}$, donc nous supposons pour l'instant que $m = \frac{d}{2}$.

L'idée principale de l'algorithme qui résout MCLM lorsque $U_h = 2$ et $U_v = 1$ est de router chaque liaison suivant un plus court chemin. Il se divise en m itérations, la i^e itération correspondant à la construction des chemins sur la i^e ligne horizontale. **Tirer une liaison** vers la droite (resp. vers la gauche) signifie router son chemin horizontalement vers la droite (resp. vers la gauche) sur la ligne courante. Notre algorithme peut être décrit de façon informelle ainsi :

- On construit les chemins ligne par ligne, du bord supérieur au bord inférieur : à chaque ligne, on obtient une nouvelle instance de MCLM ;
- Pour chaque ligne, on calcule les densités résiduelles (c'est-à-dire, les densités de la nouvelle instance de MCLM) et on met à jour les régions denses ;
- Pour chaque région dense $[\alpha, \beta]$
  1. On commence par une phase *Routage_ gauche*, où, en partant de β, on choisit séquentiellement une liaison vers la gauche et on la tire vers la gauche ;
  2. Ensuite, on procède à une phase symétrique (*Routage_ droite*) où on tire vers la droite des liaisons vers la droite.

À la fin de la i^e itération, pour chaque j, il existe une liaison dont le chemin "partiel" est arrêté en colonne j sur la i^e ligne : soit $N_{i+1,j}$ cette liaison. Alors, j peut être vu comme l'indice de la source courante de la liaison $N_{i+1,j}$ pour la nouvelle instance de MCLM obtenue à l'itération $i + 1$. On dira qu'une colonne j est **inutilisée** à une itération i si aucun chemin n'est routé vers ou à partir d'elle pendant cette itération. Donnons à présent formellement la procédure 1-2-PlusCourtsChemins qui résout MCLM.

PROCÉDURE 1-2-PlusCourtsChemins
Pour j **de** 1 **à** $|\mathcal{L}|$ **faire**
    // initialement, la colonne de la source de la liaison l_j est source(l_j)
    $N_{1,source(l_j)} := j$;
FinPour
Pour i **de** 1 **à** m **faire**
    Pour chaque région dense $[\alpha, \beta]$ **faire**
        Routage_gauche($[\alpha, \beta], i$) ;
        Routage_droite($[\alpha, \beta], i$) ;
    FinPour
    Mettre à jour les densités résiduelles ;
    // on descend les liaisons
    Router verticalement toutes les liaisons vers la prochaine ligne horizontale ;
    Pour chaque colonne inutilisée j **faire**
        $N_{i+1,j} := N_{i,j}$; *// la liaison ayant sa source sur la j^e colonne est inchangée*
    FinPour
FinPour

Les phases de routage à gauche et à droite sont symétriques, donc nous donnons seulement la première. On peut obtenir la seconde en remplaçant "α" par "β", "gauche" par "droite" et "$>$" par "$<$".

PROCÉDURE Routage_gauche($[\alpha, \beta], i$)
$j := N_{i,\beta}$; // l_j *est la liaison dont la source est en* β
α_est_atteint := **faux** ;
Tant que l_j est une liaison à gauche **et que** $\neg$ α_est_atteint **faire**
  Si $j > \alpha$ **alors** // j *est la colonne de* s'_j
    Tirer l_j vers la gauche jusqu'à ce que j soit atteint ;
    $N_{i+1,j} := j$; // l_j *devient une liaison verticale*
    $j := N_{i,j}$; // l_j *est maintenant la liaison dont la source est en* j
  Sinon
    Tirer l_j vers la gauche jusqu'à ce que α soit atteint ;
    $N_{i+1,\alpha} := j$; // *la source de* l_j *est maintenant en* α
    α_est_atteint := **vrai** ;
  FinSi
FinTantQue

Un exemple de routage donné par l'algorithme 1-2-PlusCourtsChemins est dessiné en figure 5.1.

Preuve de validité

Dans cette partie, nous prouvons par récurrence que, quand $U_h = 2$, $U_v = 1$ et $m = \frac{d}{2}$, l'algorithme 1-2-PlusCourtsChemins fournit une solution admissible pour MCLM où chaque liaison est routée suivant un plus court chemin. Considérons la i^e itération, et supposons que, au début de cette itération, on a une *instance initiale*, c.-à-d., une instance de MCLM dans un canal dense de densité $d - 2(i-1)$ et avec $m - i + 1$ lignes (évidemment, c'est vrai pour $i = 1$). Alors, nous montrons que, à la fin de l'itération, on obtient une *instance réduite*, c.-à-d., une instance valide de MCLM dans un canal dense de densité $d - 2i$ et avec $m - i$ lignes.

Fait 5.1. *Lors d'une phase de routage à gauche, aucune liaison dont la source courante est dans une région dense* $[\alpha, \beta]$ *n'est routée de telle façon que sa nouvelle source n'est plus dans* $[\alpha, \beta]$*, puisque on arrête son chemin soit lorsque* α *est atteint, soit avant que* α *ne soit atteint.*

Fait 5.2. *Lors d'une phase de routage à gauche, aucune liaison à gauche* (s_i, s'_i) *n'est routée de façon à devenir une liaison à droite lors de la prochaine itération, puisque on arrête son chemin soit lorsque* i *est atteint, soit avant que* i *ne soit atteint.*

Fait 5.3. *Pour chaque liaison sélectionnée lors d'une phase de routage à gauche, le chemin associé ne traverse que des bandes verticales qui doivent être traversées. Cela vient du fait que seules des liaisons à gauche sont tirées*

vers la gauche, et, d'après le fait 5.2*, aucune liaison à gauche ne devient une liaison à droite après avoir été tirée.*

Fait 5.4. *Des variantes symétriques des faits* 5.1*,* 5.2 *et* 5.3 *sont également vraies pour la phase de routage à droite.*

Lemme 5.2. *Le routage construit à l'itération i est valide.*

Démonstration. Considérons la phase de routage à gauche (les deux phases étant indépendantes et symétriques). D'après (5.1) et le lemme 5.1, étant donnée une région dense $[\alpha, \beta]$, la liaison l_c dont la source courante se trouve sur β est une liaison à gauche, donc on peut la tirer vers la gauche. Son chemin s'arrête en α (si $c \leq \alpha$) ou en c (si $\alpha < c \leq \beta$), où c est la colonne de son puits. Dans le dernier cas, $d_{c-1} = d_c$, donc nous savons d'après le lemme 5.1 que l'on peut continuer la phase de routage à gauche avec une liaison à gauche ayant sa source en c, et ainsi de suite jusqu'à ce que α soit atteint. □

Lemme 5.3. *À l'itération i, la densité de l'instance réduite est $d - 2i$.*

Démonstration. Lors de la phase de routage à gauche, étant donnée la région dense courante $[\alpha, \beta]$, on débute en β, puis on tire des liaisons vers la gauche jusqu'à ce qu'on atteigne α. D'après la preuve du lemme 5.2, chaque liaison sélectionnée durant cette phase commence là où la précédente finit. Donc, d'après les faits 5.3 et 5.4, la densité de chaque bande verticale située dans une région dense est diminuée de 2 quand la i^e itération finit. D'après les faits 5.1 et 5.4, les densités des autres bandes verticales demeurent inchangées, donc aucune densité résiduelle n'excède $d - 2i$. □

Le lemme 5.3 implique que, à la fin de la m^e itération, la densité résiduelle est $d - 2m = 0$: par conséquent, chaque chemin a atteint la colonne du puits correspondant, et on a fini. Donc, notre algorithme fournit une solution admissible.

Lemme 5.4. *Les chemins produits par l'algorithme sont des plus courts chemins.*

Démonstration. Immédiate d'après les faits 5.3 et 5.4, aucune liaison n'étant tirée "vers le haut". □

Théorème 5.3. *Si $m = \frac{d}{2}$, l'algorithme* 1-2-PlusCourtsChemins *produit une solution admissible pour MCLM où toutes les liaisons sont routées suivant des plus courts chemins. En outre, il peut être exécuté en temps $O(m|\mathcal{L}|)$.*

Démonstration. La première partie est une conséquence directe des lemmes 5.2, 5.3 et 5.4.

Nous prouvons maintenant la deuxième partie. À chaque itération, on doit calculer les densités résiduelles. En fait, en utilisant le lemme 5.1, on calcule les densités une fois en temps $O(|\mathcal{L}|)$ (avant la première itération), et ensuite, à chaque itération, le lemme 5.3 implique qu'il suffit de diminuer d_j de deux pour chaque bande verticale v_j située dans une région dense. La mise à jour des régions denses peut être faite en temps $O(|\mathcal{L}|)$ en parcourant les bandes verticales une fois, et les phases de routage à gauche et à droite peuvent être exécutées en $O(\beta - \alpha)$ pour chaque région dense $[\alpha, \beta]$. Il y a m itérations, donc notre algorithme peut être exécuté en $O(m|\mathcal{L}|)$, ce qui est linéaire en la taille de la grille. □

MCLM avec une capacité paire sur chaque ligne

Dans cette section, nous montrons comment généraliser les résultats précédents au cas où U_h est pair et $U_v = 1$. D'après la section 5.3.1, une condition nécessaire pour la solubilité de l'instance est (5.8), c.-à-d., $m \geq \lceil \frac{d}{U_h} \rceil$. Par conséquent, dans la suite de la section 5.3.2, on supposera que $m = \lceil \frac{d}{U_h} \rceil$.

Soit $U_h = 2p$, $1 \leq p \leq \frac{d}{2}$. On commence par transformer la grille avec m lignes et une capacité égale à $2p$ sur chaque arête horizontale en une nouvelle grille avec mp lignes et une capacité égale à 2 sur chaque arête horizontale. La capacité de chaque arête verticale est 1 dans les deux grilles. On voit facilement que la nouvelle grille vérifie également (5.8), et on peut donc appliquer l'algorithme 1-2-PlusCourtsChemins et obtenir ainsi un ensemble admissible de plus courts chemins. On utilise ensuite cet ensemble de chemins pour construire une solution pour la grille initiale. Pour chaque i, le chemin de chaque liaison dans la grille initiale est routé à travers l'arête horizontale de la i^e ligne située dans la j^e bande verticale si et seulement si, dans la nouvelle grille, il est routé à travers une arête horizontale de cette bande verticale se trouvant sur une ligne située entre la $(i-1)p+1^e$ et la ip^e. En d'autres termes, pour chaque liaison, son chemin dans la grille initiale arrive sur (resp. repart de) la i^e ligne à l'endroit où il arrive sur (resp. repart de) la $(i-1)p+1^e$ ligne (resp. la ip^e ligne) dans la nouvelle grille.

Les deux faits suivants sont vrais dans la grille initiale, puisqu'autrement ils ne seraient pas vrais dans la nouvelle grille : deux chemins quelconques ne partagent aucune arête verticale, et les chemins que nous obtenons sont des plus courts chemins. En outre, d'après la façon dont nous regroupons les lignes de la nouvelle grille, au plus $2p$ chemins sont routés à travers chaque arête horizontale de la grille initiale (c.-à-d., au plus 2 chemins pour chacune des p arêtes horizontales correspondant dans la nouvelle grille).

5.3.3 MCLM avec une capacité impaire sur chaque ligne et unitaire sur chaque colonne

Dans cette section, nous prouvons que le cas où $U_v = 1$ et U_h est un nombre impair plus grand que 2 peut être réduit au cas de la section 5.3.2. Considérons un routage : étant donnée n'importe quelle arête horizontale, un nombre pair de chemins est routé à travers elle. En effet, pour chaque chemin routé à travers elle "de gauche à droite", il y a nécessairement un chemin routé à travers elle "de droite à gauche", puisque $U_v = 1$ et il y a autant de chemins que de colonnes. Donc, on ne modifie pas la solubilité de l'instance si on remplace U_h (impair) par $U_h - 1$ (pair). Par conséquent, soit $m \geq \lceil \frac{d}{U_h - 1} \rceil$ et on peut alors appliquer les résultats de la section 5.3.2, soit il n'y a aucune solution.

5.3.4 MCLM avec une capacité au moins 2 sur chaque colonne

Les sections 5.3.2 et 5.3.3 règlent le cas où $U_v = 1$ et $U_h \geq 2$. Dans cette section, on traite le cas où $U_v \geq 2$. En fait, nous verrons que ce cas peut se réduire au cas où $U_v = 2$, car, dans le routage que nous construirons, au plus deux chemins partageront une arête verticale donnée. Par conséquent, nous supposons dans toute cette section que $U_v = 2$.

MCLM avec une capacité 2 sur chaque colonne et 1 sur chaque ligne

$U_h = 1$, donc (5.8) devient $m \geq d$. Par conséquent, on suppose que $m = d$. D'après le lemme 5.1, d est pair, donc m est pair. On procède successivement aux phases de routage à gauche et à droite sur les $2i - 1^e$ et $2i^e$ lignes respectivement, pour i de 1 à $\frac{m}{2}$. Alors, des arguments semblables à ceux utilisés en section 5.3.2 montrent que l'on obtient un ensemble admissible de plus courts chemins, et donc une solution optimale pour MCLM. La seule différence notable avec le cas étudié en section 5.3.2 se produit, pour chaque i, après la fin de la phase de routage à gauche sur la $2i - 1^e$ ligne : pour chaque région dense $[\alpha, \beta]$, il y a une liaison à droite dont la source se trouve sur la colonne α, donc une liaison à droite *et* une liaison à gauche (ou une liaison à gauche qui est devenue une liaison à droite après la phase de routage sur la $2i - 1^e$ ligne) seront routées à travers l'arête verticale de la colonne α se trouvant entre les $2i - 1^e$ et $2i^e$ lignes.

MCLM avec une capacité 2 sur chaque colonne et au moins 2 sur chaque ligne

Nous empruntons quelques idées à la fin de la section 5.3.2. On transforme la grille initiale (qui satisfait la condition nécessaire (5.8)) en une nouvelle

grille où les arêtes verticales sont également pondérées par 2, les arêtes horizontales par 1, et ayant mU_h lignes. Si mU_h est impair alors $mU_h > d$, puisque d est pair et (5.8) est vérifié (c.-à-d., $mU_h \geq d$). Donc, dans ce cas, la nouvelle grille pourrait n'avoir que d lignes seulement. Cependant, par souci de simplicité, on suppose qu'elle a mU_h lignes et que, de la d^e ligne à la ${mU_h}^e$ ligne, les liaisons sont routées verticalement. On utilise les résultats concernant le cas $U_v = 2$ et $U_h = 1$ pour obtenir une solution optimale pour MCLM dans cette nouvelle grille : ensuite, pour chaque i, le chemin de chaque liaison dans la grille initiale arrive sur (resp. repart de) la i^e ligne à l'endroit où elle arrive sur (resp. repart de) la $(i-1)U_h + 1^e$ ligne (resp. la iU_h^e ligne) dans la nouvelle grille. Une analyse similaire à celle de la section 5.3.2 prouve que ceci fournit bien une solution pour MCLM dans la grille initiale, où toutes les liaisons sont routées suivant des plus courts chemins.

Ce dernier cas permet de conclure que notre approche résout bien MCLM en temps polynomial dans les canaux denses ayant des capacités horizontales et verticales quelconques, ce qui étend les résultats de [66], où seules des capacités unitaires étaient considérées.

5.4 Complexité de MCM et MFEMI dans les grilles

Nous montrons dans cette section que MCM et MFEMI sont $\mathcal{NP}$-difficiles dans les grilles. Plus précisément, nous montrons que MCM est $\mathcal{NP}$-difficile dans les grilles uniformes (même si tous les terminaux sont sur le bord), et dans les grilles non uniformes avec seulement deux lignes, même si tous les terminaux sont sur un bord, sont tous distincts deux à deux, et si les poids sont polynomialement bornés (section 5.4.1). Nous montrons ensuite que MFEMI est $\mathcal{NP}$-difficile au sens fort, même si tous les terminaux sont sur le bord (section 5.4.2).

5.4.1 Complexité de MCM

Nous montrons d'abord que MCM est $\mathcal{NP}$-difficile dans les grilles uniformes. Pour cela, nous faisons une réduction à partir du problème $\mathcal{NP}$-complet CouvertureParLesSommets [76] :

Données : Un graphe $H = (U, B)$, un entier $N \leq |U|$.

Problème : H admet-il une couverture par les sommets, c'est-à-dire un ensemble de sommets $U' \subseteq U$ tel que pour toute arête $(u, v) \in B$ on a $u \in U'$ ou $v \in U'$, de taille au plus N ?

Soit $\mathcal{C}$ une instance de CouvertureParLesSommets. À partir de $\mathcal{C}$, nous définissons une instance $\mathcal{M}$ de MC, le problème de décision associé à MCM (qui s'énonce ainsi : étant donné un entier N, existe-t-il une multicoupe de poids au plus N ?). Soient $U = \{u_1, \ldots, u_n\}$ et c $= 1$. La grille

$(G, \mathcal{L})$ a $2n-1$ lignes verticales et $n+2$ lignes horizontales (ce n'est *pas* une grille augmentée). On définit g_j^i comme étant le sommet se trouvant sur la i^e ligne horizontale et la j^e ligne verticale de la grille. On définit les liaisons (g_j^1, g_{j+1}^1), $j \in \{1, \ldots, 2n-2\}$, et (g_{2j}^1, g_{2j}^{n+2}), $j \in \{1, \ldots, n-1\}$. Notons que la meilleure façon de couper ces liaisons est d'enlever les *arêtes* (g_j^1, g_{j+1}^1), $j \in \{1, \ldots, 2n-2\}$, et (g_{2j}^1, g_{2j}^2), $j \in \{1, \ldots, n-1\}$; il reste alors une grille où chaque terminal est relié au reste de la grille par une seule arête (plusieurs terminaux pouvant être liés au même sommet). En outre, pour chaque arête $(u_i, u_j) \in B$, on définit la liaison (g_{2i-1}^1, g_{2j-1}^1).

Lemme 5.5. *Étant donné un entier $N \leq n$, il existe une couverture par les sommets $\hat{\mathrm{C}}$ de taille au plus N dans $\mathcal{C}$ si et seulement si $\mathcal{M}$ admet une multicoupe $\bar{\mathrm{C}}$ de poids au plus $3(n-1)+N$.*

Démonstration. D'abord, nous montrons la partie $\Rightarrow$ (*nécessité*). Supposons que l'on ait une couverture par les sommets $\hat{\mathrm{C}}$ de taille $|\hat{\mathrm{C}}| \leq N$. On sélectionne dans la coupe $\bar{\mathrm{C}}$ toutes les arêtes (g_j^1, g_{j+1}^1), $j \in \{1, \ldots, 2n-2\}$, et (g_{2j}^1, g_{2j}^2), $j \in \{1, \ldots, n-1\}$. En outre, pour chaque sommet u_j dans $\hat{\mathrm{C}}$, on sélectionne dans $\bar{\mathrm{C}}$ l'arête (g_{2j-1}^1, g_{2j-1}^2). On obtient un ensemble d'arêtes de taille $3(n-1)+|\hat{\mathrm{C}}| \leq 3(n-1)+N$.

Pour voir que $\bar{\mathrm{C}}$ est une multicoupe, il faut d'abord remarquer que toutes les liaisons (g_j^1, g_{j+1}^1), $j \in \{1, \ldots, 2n-2\}$, et (g_{2j}^1, g_{2j}^{n+2}), $j \in \{1, \ldots, n-1\}$, sont déconnectées. Il reste seulement à montrer que toutes les liaisons (g_{2i-1}^1, g_{2j-1}^1), avec (i, j) tels que $(u_i, u_j) \in B$, sont coupées. Supposons qu'il en existe une qui ne soit pas coupée, et appelons-la (g_{2r-1}^1, g_{2s-1}^1). Alors, ni (g_{2r-1}^1, g_{2r-1}^2), ni (g_{2s-1}^1, g_{2s-1}^2), n'est dans la coupe. Donc, par la construction de $\bar{\mathrm{C}}$, ni u_r ni u_s n'est dans $\hat{\mathrm{C}}$, alors que $(u_r, u_s) \in B$ puisque la liaison (g_{2r-1}^1, g_{2s-1}^1) existe : une contradiction. Ceci conclut la partie nécessité.

Nous montrons maintenant la partie $\Leftarrow$ (*suffisance*). Supposons que l'on ait une multicoupe $\bar{\mathrm{C}}$ de taille $3(n-1)+N'$, avec $N' \leq N \leq n$ et $N' \in \mathbb{Z}$. Chaque arête (g_j^1, g_{j+1}^1), $j \in \{1, \ldots, 2n-2\}$, est dans $\bar{\mathrm{C}}$ puisque ses deux extrémités définissent une liaison. En outre, puisque toutes les liaisons (g_{2j}^1, g_{2j}^{n+2}), $j \in \{1, \ldots, n-1\}$, sont coupées, $\bar{\mathrm{C}}$ contient une arête verticale de la $2j^e$ ligne verticale, pour $j \in \{1, \ldots, n-1\}$. Ainsi, il existe un sous-ensemble $\bar{\mathrm{C}}'$ de $\bar{\mathrm{C}}$ qui contient $3(n-1)$ arêtes de ce type : donc, en fait, $|\bar{\mathrm{C}}| \geq 3(n-1)$ et $N' \geq 0$. Soit $\bar{\mathrm{C}}_{\mathrm{N}'} = \bar{\mathrm{C}} \setminus \bar{\mathrm{C}}'$: on a $|\bar{\mathrm{C}}_{\mathrm{N}'}| = N'$.

Soit $(\bar{G}, \mathcal{L})$ le graphe obtenu à partir de $(G, \mathcal{L})$ en enlevant toutes les arêtes dans $\bar{\mathrm{C}}$. On peut partitionner les sommets g_{2j-1}^1 en deux ensembles : le premier, F, est l'ensemble des sommets de ce type qui peuvent, dans $(\bar{G}, \mathcal{L})$, être reliés par une chaîne à g_j^{n+2} pour un j (c.-à-d., à un sommet de la $n+2^e$ ligne horizontale de $(G, \mathcal{L})$) ; le second, $\bar{F}$, contient tous les autres sommets de ce type.

À présent, montrons que tous les sommets de F sont dans la même composante connexe de $(\bar{G}, \mathcal{L})$, c'est-à-dire que, étant donnés $g^1_{2r-1} \in F$ et $g^1_{2s-1} \in F$, il existe une chaîne reliant g^1_{2r-1} à g^1_{2s-1} dans $(\bar{G}, \mathcal{L})$. Soit p_r (resp. p_s) une chaîne (dans $(\bar{G}, \mathcal{L})$) reliant g^1_{2r-1} (resp. g^1_{2s-1}) à un sommet de la $n+2^e$ ligne horizontale de $(G, \mathcal{L})$. Si p_r et p_s ont au moins un sommet en commun ou si elles peuvent être reliées par une chaîne contenant uniquement des arêtes horizontales, la preuve est finie. Sinon, $\bar{\mathrm{C}}$ contient une arête de la j^e ligne horizontale de $(G, \mathcal{L})$, pour $j \in \{2, \ldots, n+2\}$. Plus précisément, ces $n+1$ arêtes horizontales appartiennent à $\bar{\mathrm{C}}_{\mathrm{N}'}$, et donc $|\bar{\mathrm{C}}_{\mathrm{N}'}| \geq n+1$: cela contredit le fait que $N' \leq N \leq n$.

Par conséquent, il n'existe aucune liaison $(g^1_{2r-1}, g^1_{2s-1}) \in \mathcal{L}$ avec $g^1_{2r-1} \in F$ et $g^1_{2s-1} \in F$, puisqu'autrement $\bar{\mathrm{C}}$ n'est pas une multicoupe : donc, chaque liaison a au moins un terminal dans $\bar{F}$.

En outre, pour chaque j tel que $g^1_{2j-1} \in \bar{F}$ (c'est-à-dire que g^1_{2j-1} est un sommet terminal séparé de tous les sommets de la $n+2^e$ ligne horizontale de $(G, \mathcal{L})$), il n'est pas difficile de voir qu'il existe une arête verticale de la $2j-1^e$ ligne verticale qui est dans $\bar{\mathrm{C}}$ (et, plus précisément, dans $\bar{\mathrm{C}}_{N'}$). Cela implique que $|\bar{F}| \leq |\bar{\mathrm{C}}_{\mathrm{N}'}|$ $(= N')$.

Par conséquent, on peut construire une couverture par les sommets $\hat{\mathrm{C}}$ en sélectionnant chaque sommet $u_j \in U$ tel que $g^1_{2j-1} \in \bar{F}$. $\hat{\mathrm{C}}$ est effectivement une couverture par les sommets, puisque, comme nous l'avons montré précédemment, chaque liaison (soit chaque arête dans U) a au moins une extrémité dans $\bar{F}$. $|\hat{\mathrm{C}}| = |\bar{F}| \leq N' \leq N$, donc le lemme est démontré. □

Puisque MC est dans $\mathcal{NP}$ et que la réduction présentée ci-dessus est polynomiale, le lemme 5.5 implique :

Théorème 5.4. *MC est $\mathcal{NP}$-complet dans les grilles non pondérées, même si les terminaux sont sur le bord de la grille.*

Corollaire 5.1. *MCM est $\mathcal{NP}$-difficile dans les grilles non pondérées, même si les terminaux sont sur le bord de la grille.*

Nous étudions maintenant la complexité de MCM lorsque tous les terminaux sont deux à deux distincts. Nous montrons que ce problème est $\mathcal{NP}$-difficile dans les grilles (non uniformes), même lorsque l'on impose que tous les terminaux sont sur le bord. La réduction se fait encore à partir de CouvertureParLesSommets.

Soit une instance $\mathcal{C}$ de CouvertureParLesSommets sur un graphe $G = (S, A)$. On note $n = |S|$ et $m = |A|$. On lui associe une instance $\mathcal{I}_G$ du problème MCM dans les grilles de la façon suivante :

- La grille possède deux lignes et n colonnes ;
- On associe un terminal t_i de $\mathcal{I}_G$ à chaque sommet u_i de $\mathcal{C}$. On place les terminaux sur la ligne horizontale du bas (un terminal par colonne).

On associe à chaque arête (u_i, u_j) de G une liaison (t_i, t_j) dans $\mathcal{I}_G$. On pondère $\mathcal{I}_G$ de la façon suivante :

- Chacune des $n-1$ arêtes de la ligne inférieure est pondérée par 1 ;
- Chaque arête verticale (il y en a n) est pondérée par n ;
- Chaque arête de la ligne supérieure est pondérée par $n(n+1)$.

Alors il existe une couverture par les sommets de taille au plus N dans $\mathcal{C}$ si et seulement si il existe une multicoupe de poids au plus $n(N+1)-1$ dans $\mathcal{I}_G$.

En effet, s'il existe une couverture par les sommets de taille au plus N alors en coupant les arêtes situées à la verticale des terminaux correspondant aux sommets de la couverture, ainsi que toutes les arêtes de la ligne horizontale du bas, on obtient une multicoupe de valeur $nN+(n-1) = n(N+1)-1$.

Réciproquement, s'il existe une multicoupe de valeur au plus $n(N+1)-1$, alors :

- Aucune arête de la ligne horizontale du haut n'est coupée (car $n(N+1)-1 < n(n+1)$, puisque $N \leq n$) ;
- Au plus N arêtes verticales sont coupées (car $n(N+1)-1 < n(N+1)$).

Alors, en prenant les sommets u_i associés aux terminaux t_i à la verticale desquels il y a une arête coupée, on obtient une couverture par les sommets de taille au plus N.

Une légère modification permet d'adapter l'instance $\mathcal{I}_G$ au cas où l'on impose que les terminaux sont disjoints, et d'obtenir ainsi le résultat énoncé précédemment. La grille possède alors deux lignes et $2m$ colonnes. On associe, à chaque sommet u_i de G, $deg(u_i)$ terminaux notés $t_i^1, t_i^2, \ldots, t_i^{deg(u_i)}$ dans $\mathcal{I}_G$ (on rappelle que $\sum_{i=1}^{n} deg(u_i) = 2m$). Les terminaux sont, comme précédemment, situés sur la ligne du bas, les t_i^j étant regroupés par paquets (c.-à-d., contigus) pour un même i. En outre, on les ordonne de gauche à droite, t_i^1 étant le plus à gauche parmi les t_i^j et $t_i^{deg(u_i)}$ étant le plus à droite, toujours parmi les t_i^j (pour un i donné). Entre ces deux terminaux, on trouve donc tous les t_i^j, pour j de 2 à $deg(u_i)-1$, et aucun autre terminal. On ordonne les arêtes de G comme suit : $(i,j) < (i',j') \Leftrightarrow (i < i')$ ou $(i = i'$ et $j < j')$. On associe ensuite à chaque arête de G une liaison dans $\mathcal{I}_G$, de telle façon que chaque apparition d'un sommet u_i dans une arête soit associée à un terminal t_i^j différent. On pondère ensuite la grille de la façon suivante (voir la figure 5.2) :

- Chaque arête de la forme $(t_i^{deg(u_i)}, t_{i+1}^1)$ de la ligne horizontale du bas (il y en a $n-1$) est pondérée par 1 ;
- Les autres arêtes de la ligne horizontale du bas sont pondérées par $2m(n+1)$;
- Les arêtes de la ligne horizontale du haut sont pondérées par $2m(n+1)$;
- Chaque arête à la verticale d'un terminal de la forme t_i^1 (il y en a n)

est pondérée par $2m$;
– Les autres arêtes verticales sont pondérées par 1 (il y en a $2m - n$).

On montre alors, par un raisonnement semblable au précédent, qu'il existe une couverture par les sommets de taille au plus N si et seulement si il existe une multicoupe de valeur au plus $2m(N+1)-1$: la multicoupe est formée des arêtes horizontales et verticales de valeur 1 et de N arêtes verticales de valeur $2m$; réciproquement, la couverture est formée des sommets u_i tels que l'arête de poids $2m$ située à la verticale du terminal t_i^1 est coupée. Cela conclut la preuve. D'où :

Théorème 5.5. *MCM est $\mathcal{NP}$-difficile au sens fort dans les grilles, même si tous les terminaux sont sur un des bords de la grille et sont deux à deux distincts.*

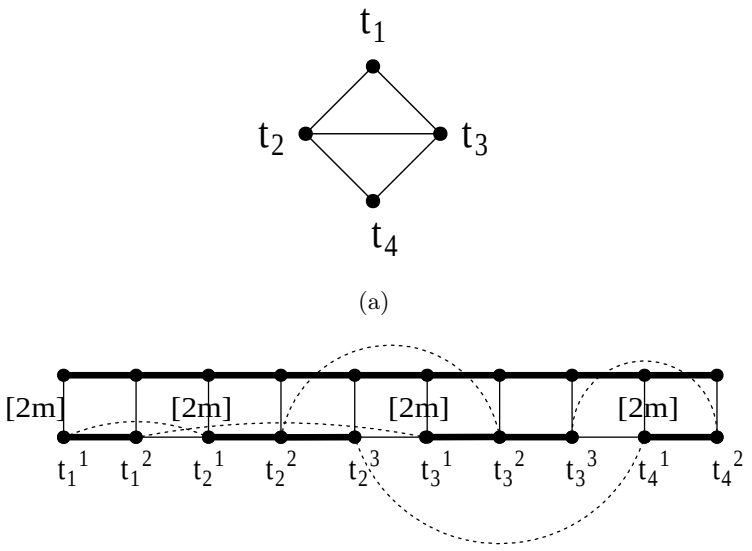


FIG. 5.2 – Une instance de COUVERTUREPARLESSOMMETS à 4 sommets et l'instance de MCM associée ($m = 5$ ici). Les arêtes en gras sont pondérées par $2m(n+1)$ (50 ici) ; les arêtes en trait fin dont le poids n'est pas précisé sont pondérées par 1. En outre, les liaisons sont dessinées en pointillés.

5.4.2 Complexité de MFEMI

Nous montrons que MFEMI est $\mathcal{NP}$-difficile au sens fort dans les grilles en effectuant une réduction à partir du problème $\mathcal{NP}$-complet au sens fort 3-PARTITION [76] :

Données : Un ensemble E de $3m$ éléments, un entier $B \in \mathbb{N}^*$ et une taille $f(e_i) \in \mathbb{N}^*$ pour tout $e_i \in E$ telle que $\frac{B}{4} < f(e_i) < \frac{B}{2}$ et telle que $\sum_{e_i \in E} f(e_i) = mB$.

Problème : Peut-on partitionner E en m ensembles disjoints $E_1, \ldots, E_m$ tels que $\forall\ 1 \leq j \leq m$, $\sum_{e_i \in E_j} f(e_i) = B$?

Notons que la condition $\frac{B}{4} < f(e_i) < \frac{B}{2}$ pour tout $e_i \in E$ implique que chaque E_j doit contenir exactement trois éléments de E. Pour notre réduction, on va utiliser une variante de ce problème dans laquelle $f(e_i) \geq 2$ pour tout $e_i \in E$. Il n'est pas difficile de voir que ce problème reste $\mathcal{NP}$-complet au sens fort, car toute instance de 3-PARTITION peut être transformée en une instance équivalente de ce problème en multipliant toutes les données, sauf m (c'est-à-dire B et $f(e_i)$ $\forall e_i \in E$), par 2.

Étant donnée une instance $\mathcal{I}$ de la variante de 3-PARTITION, on construit une instance $\mathcal{I}'$ de MFEMI comme suit (voir la figure 5.3) :

- La grille possède $m+1$ lignes et $6m$ colonnes ;
- Les arêtes horizontales de la bande verticale située au milieu de la grille (c.-à-d., entre les colonnes $3m$ et $3m+1$) sont pondérées par B, sauf celle qui est située sur la première ligne, qui est pondérée par 1 ;
- La i^e arête verticale de la première bande horizontale est pondérée par $f(e_i)$, pour $i \in \{1, \ldots, 3m\}$;
- Toutes les autres arêtes sont pondérées par mB ;
- Il y a $3m$ liaisons, et tous les terminaux sont sur la ligne du haut ;
- La source et le puits de la i^e liaison sont situés respectivement sur la i^e colonne et sur la $(3m+i)^e$ colonne.

Le principe de la réduction est que tout chemin d'une source à son puits doit traverser la bande verticale du milieu. Or, les capacités de la première bande horizontale garantissent que la somme des flots routés vaut $\sum_{e_i \in E} f(e_i)$ si et seulement si le i^e des $3m$ flots vaut $f(e_i)$ (aucune unité de flot n'étant, dans ce cas, routée à travers l'arête de poids 1 puisque $f(e_i) \geq 2$ pour tout i et que les flots sont insécables). En outre, si la somme des flots routés vaut $\sum_{e_i \in E} f(e_i)$, trois flots sont routés à travers l'arête se trouvant sur la bande verticale du milieu et sur la i^e ligne, pour $i \in \{2, \ldots, m+1\}$, et la somme de ces trois flots vaut B. Cette partition des flots fournit ainsi la partition des éléments de E. Donc, il existe une solution à $\mathcal{I}$ si et seulement si la valeur optimale de $\mathcal{I}'$ est mB. D'où :

Théorème 5.6. *MFEMI est $\mathcal{NP}$-difficile au sens fort dans les grilles, même si tous les terminaux sont du même bord et distincts deux à deux.*

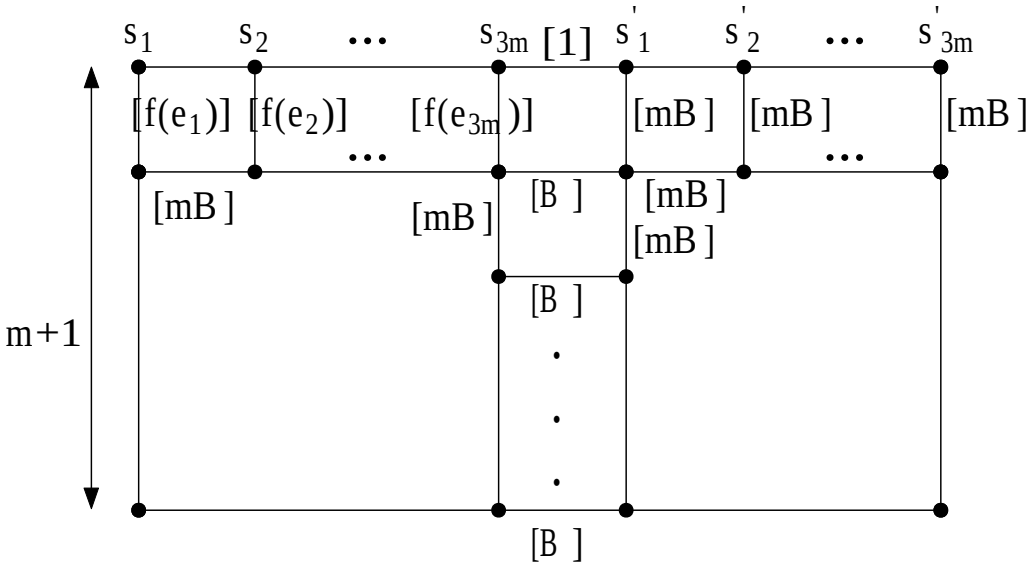


FIG. 5.3 – Instance pour la réduction de 3-PARTITION vers MFEMI.

5.5 Polynomialité de MFEM et MCM dans une famille de grilles uniformes

Dans cette section, nous élaborons des algorithmes polynomiaux pour résoudre MFEM et MCM dans une famille de grilles uniformes bilatérales associée au cas particulier de CDA étudié par Frank [69] (ces résultats ont fait l'objet d'une publication en revue [19]).

Plus précisément, dans toute cette section, nous supposerons, comme dans [66, 162], que les grilles uniformes (c'est-à-dire, dont toutes les arêtes sont pondérées par le même entier c) que l'on étudie sont **augmentées**, c'est-à-dire que chaque terminal est relié à la grille par une unique arête pondérée par la capacité c, à moins que l'on précise explicitement le contraire. Cette hypothèse peut être faite sans perte de généralité, puisqu'une grille uniforme qui ne la vérifie pas peut aisément être transformée en une grille équivalente qui la vérifie. En effet, pour chaque liaison (s_i, s'_i), il suffit de remplacer s_i et s'_i par $\min(deg(s_i), deg(s'_i))$ nouveaux terminaux (avec $deg(s_i), deg(s'_i) \in \{2, 3, 4\}$) reliés par une unique arête de poids c à s_i et s'_i respectivement, et la liaison (s_i, s'_i) par $\min(deg(s_i), deg(s'_i))$ nouvelles liaisons "parallèles" définies sur ces $2\min(deg(s_i), deg(s'_i))$ nouveaux terminaux, pour tout i.

En outre, comme cela a été évoqué précédemment, on se concentrera sur des problèmes d'optimisation associés à CDA, le problème étudié par Frank dans [69] et [112, pp. 47-100]. Il suppose dans ces deux articles que les grilles sont bilatérales. Or, comme nous l'avons déjà mentionné, le problème

d'optimisation associé à CDA est MLCDA, qui est lui-même équivalent (dans les grilles) au problème MCDA défini sur une grille où chaque source s_i (resp. puits s'_i) est relié(e) au reste de la grille par une unique arête a_i (resp. a'_i). L'hypothèse faite par Frank est alors équivalente (pour les problèmes d'optimisation MCDA, MFEM et MCM) au fait de considérer que chaque terminal est relié à un sommet d'un bord horizontal et qu'au plus un terminal peut être relié à un sommet donné. On fera référence à ce cas comme étant le cas *bilatéral augmenté*. En section 5.5.3, on généralisera à MFEM nos résultats concernant MCDA, et on supposera donc que toutes les arêtes, y compris a_i et a'_i pour $i \in \{1, \ldots, |\mathcal{L}|\}$, sont pondérées par $c \geq 2$.

On peut remarquer que, d'une part, le corollaire 5.1 montre que MCM est $\mathcal{NP}$-difficile dans les grilles augmentées où les terminaux sont reliés à des sommets des bords horizontaux, si plusieurs terminaux peuvent être reliés au même sommet (en fait, même si seulement 5 terminaux peuvent être reliés au même sommet, puisque CouvertureParLesSommets reste $\mathcal{NP}$-difficile dans les graphes de degré maximum 3). D'autre part, nous montrons dans les sections 5.5.1, 5.5.2 et 5.5.3 que MCM et MFEM sont polynomiaux dans les grilles bilatérales augmentées (c'est-à-dire si au plus un terminal peut être relié à chaque sommet).

Nos résultats sont organisés de la façon suivante. Nous résolvons d'abord MCDA (section 5.5.1), puis MCM (section 5.5.2), et enfin montrons comment utiliser ces résultats pour résoudre MFEM (section 5.5.3) et MFEMI (section 5.5.4). Les sections 5.5.5 à 5.5.8 sont ensuite consacrées aux détails de certaines preuves et algorithmes.

5.5.1 Résolution de MCDA

Rappelons d'abord que, dans une grille bilatérale augmentée, chaque terminal est relié par une unique arête à un sommet de la ligne inférieure ou supérieure, et au plus un terminal peut être relié à un sommet donné. En fait, on considérera qu'on relie par une arête un et un seul sommet de degré 1 à chaque sommet des lignes inférieure et supérieure. Au plus un terminal est placé sur chacun de ces sommets de degré 1, que l'on désignera (par abus de langage) par sommets des bords horizontaux (les sommets auxquels les sommets des bords horizontaux sont reliés étant simplement appelés sommets inférieurs et supérieurs). Deux terminaux sont du même bord s'ils sont reliés à des sommets du même bord. La colonne d'un terminal désigne la colonne du sommet de la grille auquel ce terminal est relié.

Étant donnée une instance de MLCDA dans une grille bilatérale augmentée (c'est-à-dire une instance de MCDA, puisque ces deux problèmes sont équivalents dans ce cas), si toutes les liaisons sont verticales, alors elles peuvent toutes être routées verticalement. La multicoupe correspondante est trivialement obtenue en coupant chaque liaison *à sa source*, c.-à-d., en sélec-

tionnant chaque arête a_i, $i \in \{1, \ldots, |\mathcal{L}|\}$, dans la coupe : dans ce cas, on a $Opt(\text{MCDA}) = Opt(\text{MFEM}) = |\mathcal{L}|$ (dans le cas où c ≥ 2, on a $Opt(\text{MFEM})$ $= Opt(\text{MCM}) = |\mathcal{L}|\text{c}$). Afin d'éviter le cas trivial, on supposera donc dans la suite de ce chapitre qu'il y a au moins une liaison non verticale. Notons que, soit (5.5) est vérifié, soit il suffit d'enlever une seule liaison (une liaison dont la source ou le puits est lié(e) à un coin de la grille, par exemple) pour qu'il soit vérifié. D'après le théorème 5.1, on a :

Proposition 5.2. *Si $(G, \mathcal{L})$ vérifie $m \geq d$ alors*
- $Opt(\text{MCDA}) = |\mathcal{L}|$ *si $m > d$ et il existe un sommet non terminal sur un bord horizontal ;*
- $Opt(\text{MCDA}) = |\mathcal{L}|$ *si* (5.4), (5.5) *ou* (5.6) *est vérifié ;*
- $Opt(\text{MCDA}) = |\mathcal{L}| - 1$ *sinon.*

La proposition 5.2 règle le cas où (5.3) est vérifié. Notons en outre que (5.4), (5.5) et (5.6) peuvent être testés en $O(n)$. *Dans la suite de la section 5.5.1, nous supposons que $(G, \mathcal{L})$ vérifie $m < d$.* Notre approche consiste à enlever suffisamment de liaisons pour obtenir une grille qui vérifie (5.3) (on ne considère pas d'autres conditions nécessaires pour le moment). Une formulation équivalente du problème consiste à choisir dans $\mathcal{L}$ autant de liaisons que possible sans excéder une densité de m (voir les contraintes (5.9) ci-dessous). Ce problème peut être formulé ainsi :

$$
\textbf{(IPL)} \left| \begin{array}{lll}
\max & \displaystyle\sum_{i=1}^{|\mathcal{L}|} x_i & \\
\text{t. q.} & \displaystyle\sum_{i \text{ t. q. } l_i \text{ traverse } v_j} x_i \leq m & \forall j \in \{1, \ldots, n-1\} \quad (5.9) \\
& x_i \in \{0, 1\} & \forall i \in \{1, \ldots, |\mathcal{L}|\} \quad (5.10)
\end{array} \right.
$$

où la variable x_i est égale à 1 ssi la liaison $l_i = (s_i, s'_i)$ est sélectionnée. (PL), la relaxation continue de (IPL), est obtenue en remplaçant les contraintes (5.10) par

$$x_i \leq 1, \forall i \in \{1, \ldots, |\mathcal{L}|\} \tag{5.11}$$

et $x_i \geq 0$, $\forall i \in \{1, \ldots, |\mathcal{L}|\}$.

Lemme 5.6. M*, la matrice définie par le membre de gauche des contraintes* (5.9)*, est totalement unimodulaire.*

Démonstration. Les bandes verticales traversées par chaque liaison étant consécutives, il n'y a qu'une suite de 1 consécutifs sur chaque colonne de M. Ainsi, M est une matrice d'intervalle [149, p. 279] : donc, M est totalement unimodulaire. □

Par conséquent, nous savons que, m étant entier, toute solution de base de (PL) est entière : ainsi, la résolution de (IPL) est un problème polynomial. En fait, nous verrons en section 5.5.5 que (IPL) peut être résolu comme un problème d'admission d'appels sur une chaîne par un algorithme combinatoire efficace [1]. Étant donnée une solution optimale x^* quelconque pour (IPL), soit $\mathcal{L}^-$ l'ensemble des liaisons sélectionnées dans x^*, c.-à-d., l'ensemble des liaisons l_i telles que $x_i^* = 1$. Une liaison de $\mathcal{L}$ sera dite *enlevée* si elle n'appartient pas à $\mathcal{L}^-$. $(G, \mathcal{L}^-)$ a une densité égale à m, puisque, dans toute solution optimale pour (IPL), au moins une des contraintes (5.9) est saturée. Soit N^* le nombre de liaisons dans $(G, \mathcal{L}^-)$: on a $N^* = Opt(IPL)$. $(G, \mathcal{L}^-)$ vérifie (5.3), et, de nouveau, soit (5.5) est vérifié ou le retrait d'une seule liaison est suffisant pour qu'il soit vérifié, donc

$$Opt(\text{MCDA}) \in \{N^* - 1, N^*\} \tag{5.12}$$

Mais alors, est-ce que les N^* liaisons sélectionnées par (IPL) peuvent être reliées par des chemins disjoints par les arêtes ? De manière équivalente, est-ce que (IPL) admet toujours une solution optimale qui vérifie (5.4), (5.5) ou (5.6) ? La figure 5.4 montre un exemple où $N^* = 8$, mais $Opt(\text{MCDA}) = N^* - 1 = 7$: la seule solution admissible avec 8 liaisons est obtenue en enlevant (s_3, s'_3) et (s_8, s'_8), mais elle ne satisfait aucune des trois conditions (5.4), (5.5) et (5.6).

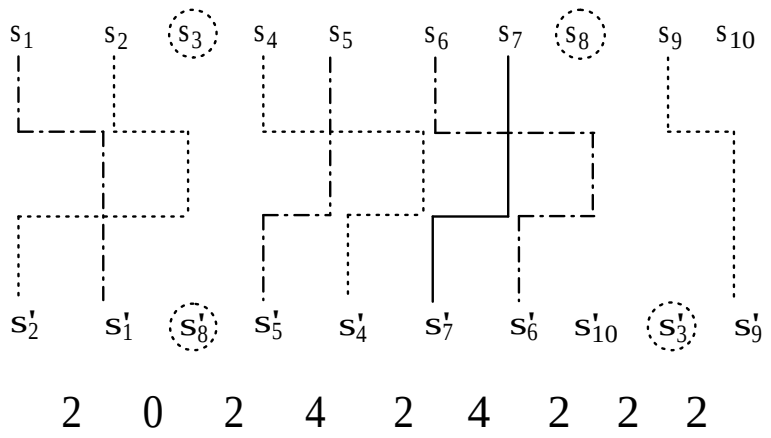


FIG. 5.4 – Une instance de MCDA où $d = 4$ (en gras), $|\mathcal{L}| = n = 10$, $m = 2$, $N^* = 8$ ((s_3, s'_3) et (s_8, s'_8) sont enlevées) et $Opt(\text{MCDA}) = 7$ (les 7 chemins sont en pointillés).

Généralement, il peut exister plusieurs solutions optimales pour (IPL), et nous devons déterminer si l'une d'entre elles satisfait (5.4), (5.5) ou (5.6). Montrons d'abord que la situation est beaucoup plus simple quand m est impair.

Proposition 5.3. *Si m est impair, alors toute grille qui vérifie $m = d$ vérifie* (5.5).

Démonstration. Supposons que (5.5) ne soit pas vérifié. Alors, tout sommet d'un bord horizontal situé à gauche (resp. à droite) de la bande verticale saturée la plus à gauche v_G (resp. la plus à droite v_D) est un terminal. Puisque dans une grille pleine toutes les densités sont paires (voir (5.2)), la même chose est vraie pour les deux sous-graphes situés respectivement à gauche de v_G et à droite de v_D. Par conséquent, d_G et d_D, les densités respectives de v_G et de v_D, sont paires. Mais, puisque v_G et v_D sont saturées, $d_G = d_D = m$. m étant impair, on obtient une contradiction. □

Puisque $(G, \mathcal{L}^-)$ vérifie $m = d$, la proposition 5.3 implique immédiatement :

Corollaire 5.2. *On a Opt*(MCDA) $= N^*$ *quand m est impair et $(G, \mathcal{L})$ vérifie $m < d$.*

On doit encore traiter le cas général. Nous montrons le théorème suivant :

Théorème 5.7. *Quand la grille $(G, \mathcal{L})$ vérifie $m < d$, une solution optimale pour* MCDA *peut être trouvée en résolvant* $\mathrm{O}(n^2)$ *programmes linéaires continus.*

Démonstration. Dans cette preuve, nous ne ferons pas la distinction entre une solution optimale x^* pour (IPL) et la grille obtenue en enlevant toutes les liaisons non sélectionnées dans cette solution (c.-à-d., toutes les liaisons l_i telles que $x_i^* = 0$). Si m est impair, alors d'après la proposition 5.3, résoudre un seul programme linéaire suffit. De même, si $(G, \mathcal{L}^-)$ vérifie (5.4), (5.5) ou (5.6), alors on a Opt(MCDA) $= N^*$, et le théorème est donc démontré. Autrement, nous recherchons d'abord une solution vérifiant (5.5). Soit v_G (resp. v_D) la bande verticale de densité m la plus à gauche (resp. la plus à droite) dans $(G, \mathcal{L})$, et soit l_g (resp. l_d) la liaison la plus D-longue (resp. la plus G-longue) traversant v_G (resp. v_D). Alors, il existe une solution optimale pour (IPL) qui vérifie (5.5) si et seulement si, l_g *ou* l_d *étant enlevée* (c.-à-d., $x_g = 0$ *ou* $x_d = 0$), (IPL) a toujours une valeur optimale égale à N^*. En effet, le sens $\Leftarrow$ est facile, et le sens $\Rightarrow$ vient du fait que, étant donnée une solution qui vérifie (5.5), on peut remplacer une liaison enlevée dont le terminal de gauche est situé à gauche de v_G (resp. dont le terminal le droite est situé à droite de v_D) par l_g (resp. par l_d), et obtenir une solution admissible de même valeur. Ainsi, nous avons seulement deux programmes linéaires à résoudre pour savoir s'il existe une solution optimale de (IPL) qui vérifie (5.5).

Si une telle solution n'existe pas, nous recherchons alors une solution vérifiant (5.6). Étant donnés deux sommets u_1 et u_2 du même bord, si u_1 (resp. u_2) est un terminal, on note l_{u_1} (resp. l_{u_2}) la liaison à laquelle il

appartient ; sinon l_{u_1} (resp. l_{u_2}) est dite *indéfinie*. Supposons sans perte de généralité que $col(u_1) < col(u_2)$. Soit $(5.9)_{u_1,u_2}$ l'ensemble de contraintes (5.9) où m est remplacé par $m-1$ pour $j \in \{col(u_1), \ldots, col(u_2)-1\}$: on a ainsi $\sum_{i \text{ t. q. } l_i \text{ traverse } v_j} x_i \leq m-1, \forall j \in \{col(u_1), \ldots, col(u_2)-1\}$, et $\sum_{i \text{ t. q. } l_i \text{ traverse } v_j} x_i \leq m, \forall j \in \{1, \ldots, col(u_1)-1\} \cup \{col(u_2), \ldots, n-1\}$. Soit $(IPL(u_1, u_2))$ le PL obtenu à partir de (IPL) en remplaçant les contraintes (5.9) par $(5.9)_{u_1,u_2}$. Ces contraintes $(5.9)_{u_1,u_2}$ garantissent l'absence de bande verticale saturée entre u_1 et u_2. En outre, pour tout (u_1, u_2), la matrice associée au membre de gauche des contraintes $(5.9)_{u_1,u_2}$ est M (comme dans (5.9)). De façon évidente, il existe une solution optimale de (IPL) qui vérifie (5.6) si, l_{u_1} *et* l_{u_2} *étant enlevées (ou indéfinies)*, on a $Opt(IPL(u_1, u_2)) = Opt(IPL) = N^*$. L'idée est alors de résoudre $(IPL(u_1, u_2))$ pour chaque paire (u_1, u_2) telle que u_1 et u_2 sont du même bord. Notons qu'il faut le faire pour les deux bords horizontaux. Ainsi, nous avons à résoudre au plus $O(n^2)$ programmes linéaires pour décider si (IPL) admet une solution optimale qui vérifie (5.6).

Enfin, en utilisant les idées ci-dessus, nous pouvons vérifier s'il y a une solution optimale vérifiant (5.4) en résolvant $O(n)$ programmes linéaires (en effet, il y a seulement $O(n)$ paires (u_1, u_2) à considérer dans ce cas). Par conséquent, nous avons $O(\max(2, n, n^2))$ programmes linéaires à résoudre pour décider si $Opt(\text{MCDA})$ est égal à N^* ou non. □

Nous verrons en section 5.5.5 que $n = O(|\mathcal{L}|)$, et donc résoudre $O(|\mathcal{L}|^2)$ programmes linéaires suffit. Enfin, une fois qu'on a calculé $Opt(\text{MCDA})$ et décidé quelles liaisons doivent être routées, on peut utiliser les algorithmes donnés dans [69] ou [162] pour construire effectivement les chemins disjoints par les arêtes.

5.5.2 Résolution de MCM

Dans cette partie, nous donnons un algorithme polynomial pour résoudre MCM dans les grilles uniformes bilatérales augmentées. Nous commençons par présenter quelques notions et résultats préliminaires.

Soit $(G, \mathcal{L})$ (avec $G = (S, A)$) une grille bilatérale augmentée et soit $\mathcal{I} = (G, \mathcal{L}, U, D)$ une instance du problème de multiflot avec demandes. Rappelons que, pour chaque liaison $l_i = (s_i, s'_i)$, a_i et a'_i sont les arêtes incidentes à s_i et à s'_i respectivement. Évidemment, une condition nécessaire de solubilité pour $\mathcal{I}$ est que $\min(U(a_i), U(a'_i)) \geq D((s_i, s'_i))$ pour chaque $i \in \{1, \ldots, |\mathcal{L}|\}$. Ainsi, supposons que cette condition soit vérifiée : alors, résoudre le problème de multiflot avec demandes sur $(G, \mathcal{L})$ équivaut à le résoudre sur la grille obtenue à partir de $(G, \mathcal{L})$ en contractant chaque a_i et chaque a'_i en un seul sommet (et dont l'ensemble d'arêtes est donc $A = A \setminus \bigcup_{i \in \{1,\ldots,|\mathcal{L}|\}} \{a_i, a'_i\}$). *Pour ce problème*, nous pouvons alors supposer que $(G, \mathcal{L})$ n'est pas une grille augmentée. Nous utiliserons le théorème 5.2

dans les sections 5.5.2 et 5.5.3. Pour l'instant, nous avons besoin du résultat suivant concernant la condition de coupe dans les grilles non augmentées.

Lemme 5.7. *Soit $(G, \mathcal{L})$ (avec $G = (S, A)$) une grille bilatérale non augmentée où*
(a) *toutes les arêtes horizontales ont la même capacité U_h,*
(b) *toutes les arêtes verticales ont la même capacité $U_v \geq \max_{l \in \mathcal{L}} D(l)$, sauf celles qui sont sur la colonne la plus à droite ou sur la colonne la plus à gauche, dont les capacités sont au plus U_v,*
(c) *toute bande horizontale h_j vérifie $\sum_{a \text{ est une arête de } h_j} U(a) \geq \sum_{i=1}^{|\mathcal{L}|} D(l_i)$,*
(d) *soit $U_h = U_v$, soit chaque arête verticale de la colonne la plus à gauche et de la colonne la plus à droite est aussi pondérée par U_v.*

Alors, la condition de coupe est vérifiée par tout $X \subseteq S$ si et seulement si elle est vérifiée par tout $X \subseteq S$ tel que $\delta_G(X)$ est une bande verticale de la grille.

La preuve du lemme 5.7 est donnée en section 5.5.6. Dans la suite, nous verrons que les grilles que nous aurons à considérer satisferont toujours les hypothèses du lemme 5.7, et que l'on pourra donc appliquer le théorème 5.2 si et seulement si la condition de coupe est vérifiée sur chaque bande verticale, c.-à-d., pour $U_h = 1$, si et seulement si $m \geq d$.

Nous pouvons à présent décrire l'algorithme résolvant MCM. Rappelons tout d'abord que, puisque nous supposons que toutes les arêtes sont pondérées par le même entier, résoudre MCM équivaut à trouver un *ensemble d'arêtes de cardinalité minimum* dont la suppression sépare s_i de s'_i pour chaque liaison l_i (ainsi, dans la suite de la section 5.5.2, nous supposerons que $c = 1$, à moins qu'une valeur différente ne soit explicitement donnée). Nous résolvons MCM dans le cas bilatéral augmenté en utilisant une approche basée sur une relation de dualité. Nous commençons par proposer une formulation en programme linéaire, et nous montrons comment cela fournit une solution admissible pour MCM. Ensuite, nous montrons qu'il existe un multiflot *continu* ayant la même valeur que cette solution particulière. Soient y_j, $j \in \{1, \ldots, n-1\}$, les variables duales associées aux contraintes (5.9) et soient w_i, $i \in \{1, \ldots, |\mathcal{L}|\}$, celles associées aux contraintes (5.11). Le programme linéaire dual de (PL) est donné par :

$$
(\boldsymbol{PC}) \left| \begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{|\mathcal{L}|} w_i + m \sum_{j=1}^{n-1} y_j & \\
\text{t. q.} & \displaystyle w_i + \sum_{j \text{ t. q. } l_i \text{ traverse } v_j} y_j \geq 1 & \forall i \in \{1, \ldots, |\mathcal{L}|\} \quad (5.13) \\
& y_j \geq 0 & \forall j \in \{1, \ldots, n-1\} (5.14) \\
& w_i \geq 0 & \forall i \in \{1, \ldots, |\mathcal{L}|\} \quad (5.15)
\end{array} \right.
$$

Lemme 5.8. (PC) *admet une solution optimale qui définit une multicoupe.*

Démonstration. D'après les contraintes (5.13), puisque la fonction objectif de (PC) doit être minimisée et qu'elle n'a que des coefficients positifs, n'importe quelle solution optimale vérifie $y_j \leq 1$ pour tout j, et $w_i \leq 1$ pour tout i. D'après le lemme 5.6, la matrice des contraintes de (PC) est totalement unimodulaire. Ainsi, si on considère seulement les solutions de base, on peut remplacer les contraintes (5.14) et (5.15) par $y_j \in \{0,1\}$ pour tout j et $w_i \in \{0,1\}$ pour tout i respectivement, et obtenir un programme linéaire en nombres entiers dont toute solution définit une multicoupe particulière C de valeur $\sum_{i=1}^{|\mathcal{L}|} w_i + m \sum_{j=1}^{n-1} y_j$, donnée par (voir la figure 5.5) :

- $w_i = 1 \Leftrightarrow a_i \in$ C;
- $y_j = 1 \Leftrightarrow v_j$ *est dans* C, c.-à-d., toutes les arêtes de v_j appartiennent à C.

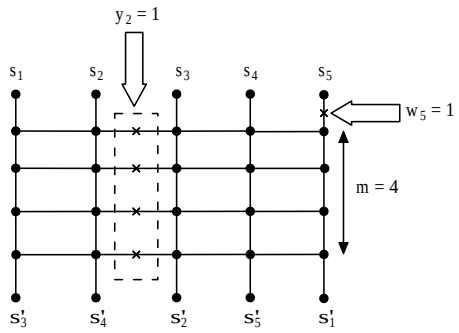


FIG. 5.5 – Une coupe donnée par (PC).

C est effectivement une multicoupe puisque, d'après (5.13), pour chaque liaison $l_i = (s_i, s'_i)$, soit $w_i = 1$ et donc a_i est dans C (alors, s_i est séparé de la grille et donc de s'_i), soit il existe un j tel que l_i traverse v_j et $y_j = 1$ (alors, il existe une bande verticale située entre s_i et s'_i dont les arêtes (horizontales) sont dans C). □

Si $(G, \mathcal{L})$ vérifie $m \geq d$, alors par convention nous écrirons N^* pour $|\mathcal{L}|$. Soit (w^*, y^*) une solution optimale entière pour (PC). D'après la relation de dualité existant entre les programmes linéaires (PL) et (PC), nous avons $Opt(PL) = Opt(PC) = N^* = \sum_{i=1}^{|\mathcal{L}|} w_i^* + m \sum_{j=1}^{n-1} y_j^*$. Ainsi nous pouvons obtenir, en temps polynomial, une solution admissible pour MCM qui contient N^* arêtes, c.-à-d., qui est optimale parmi toutes les multicoupes associées à des solutions entières de (PC). Mais il peut exister d'autres types de multicoupes, et nous devons montrer que cette solution est également optimale pour MCM, c.-à-d., qu'il existe toujours une multicoupe optimale de ce type : d'après la section 5.5.1, la valeur optimale de MCDA peut être $N^* - 1$ (voir la figure 5.4), ce qui laisse un écart d'une unité entre Opt(MCDA) et $Opt(PC)$.

Lemme 5.9. *Dans les grilles bilatérales augmentées non pondérées, on peut router un multiflot continu de valeur N^*.*

Démonstration. On considère le problème de multiflot avec demandes décrit en section 5.2. On construit une instance $\mathcal{I}$ de ce problème pour prouver le lemme 5.9 : l'idée est de travailler sur $(G, \mathcal{L}^-)$. $(G, \mathcal{L}^-)$ a N^* liaisons, et on a $U(a) = 1$ pour chaque arête a. On définit les demandes comme étant $D(l) = 1$ pour chaque liaison l dans $\mathcal{L}^-$, et on considère $\mathcal{I} = (G, \mathcal{L}^-, U, D)$. $(G, \mathcal{L}^-)$ a une densité égale à m, donc la condition de coupe est vérifiée par chaque bande verticale. En outre, $\mathcal{I}$ vérifie (a), (b), (c) et (d), et donc, d'après le lemme 5.7, $(G, \mathcal{L}^-)$ vérifie la condition de coupe ; par conséquent, le lemme 5.9 découle de la première partie du théorème 5.2. □

Si toutes les arêtes sont pondérées par un unique entier $\mathrm{c} \geq 2$, alors la valeur optimale de (PC) est $N^*\mathrm{c}$, et on peut router un multiflot continu de cette valeur : on définit $U(a) = \mathrm{c}$ pour chaque arête a et $D(l) = \mathrm{c}$ pour chaque liaison l dans $(G, \mathcal{L}^-)$, et on applique le lemme 5.7. La valeur de tout multiflot admissible étant au plus la valeur de toute multicoupe, le lemme 5.9 implique qu'il n'y a aucun saut d'intégrité pour MCM, et donc :

Corollaire 5.3. *Dans le cas bilatéral augmenté, une solution optimale pour MCM est obtenue en résolvant* (PC).

Nous donnons en section 5.5.8 une preuve alternative (et constructive) du lemme 5.9. En outre, en section 5.5.5, nous proposons un algorithme combinatoire efficace pour résoudre (PC) et donc, d'après le corollaire 5.3, pour calculer une multicoupe optimale.

5.5.3 Résolution de MFEM

Dans cette section, nous résolvons MFEM dans le cas bilatéral augmenté, *sous l'hypothèse que* $\mathrm{c} \geq 2$ (la section 5.5.1 traite le cas $\mathrm{c} = 1$). Rappelons que $N^* = Opt(PL)$ si $m < d$, et $N^* = |\mathcal{L}|$ sinon. Le résultat principal de cette section est le théorème suivant :

Théorème 5.8. *Si* (5.4) *n'est pas vérifié, si* c *est impair, si* $|\mathcal{L}| = n$ *et si* $d \leq m < \left\lceil \frac{dc}{\mathrm{c}-1} \right\rceil$, *alors* $Opt(\mathrm{MFEM}) = |\mathcal{L}|\mathrm{c} - 1$ *; Sinon* $Opt(\mathrm{MFEM}) = N^*\mathrm{c}$.

La preuve du théorème 5.8 est donnée en section 5.5.7, où les différents cas sont traités dans quatre lemmes (voir le tableau 5.1 dans cette section). Les détails de la preuve montrent également que résoudre MFEM quand $\mathrm{c} \geq 2$ nécessite simplement de trouver $(G, \mathcal{L}^-)$, c.-à-d., de résoudre un seul programme linéaire. Ceci contraste avec MCDA, pour lequel le théorème 5.7 prouve que, dans le pire cas, on doit résoudre $O(|\mathcal{L}|^2)$ programmes linéaires. En outre, on peut noter que la valeur optimale de MFEM quand $\mathrm{c} \geq 2$ est égale à $N^*\mathrm{c}$ si $m < d$ ou si m est assez grand (c.-à-d., si $m \geq \lceil \frac{dc}{\mathrm{c}-1} \rceil$) ; à

l'inverse, la valeur optimale de MCDA n'est pas toujours égale à N^* quand $m < d$, et n'est jamais égale à $|\mathcal{L}|$ lorsque $m \geq d$, que (5.4) n'est pas vérifié et que $|\mathcal{L}| = n$ (même si m est très grand).

Il faut remarquer que, bien que ce problème soit polynomial, sa résolution exacte à l'aide d'un logiciel comme CPLEX reste problématique. Par exemple, considérons une grille bilatérale augmentée avec 6 colonnes, 6 liaisons, 7 lignes, une capacité uniforme de 5 et où, sur la i^e colonne, se trouve la source de la $7 - i^e$ liaison et le puits de la i^e liaison pour tout i. Alors, d'après le théorème 5.8, la valeur optimale de MFEM est $6 \times 5 - 1 = 29$, car $6 \leq 7 < \lceil \frac{6\times 5}{4} \rceil = 8$, bien qu'il existe un multiflot continu de valeur 30. Cependant, malgré l'apparente simplicité de cette instance de taille très raisonnable, le logiciel CPLEX 9.0 [95] trouve une solution entière de valeur 29 au bout de quelques secondes seulement, mais, même au bout d'une semaine de calcul, ne parvient pas à prouver l'optimalité de cette solution (sa borne supérieure restant 30, la valeur de la relaxation continue).

5.5.4 Résolution de MFEMI

Nous traitons ici de la résolution de MFEMI dans les grilles uniformes bilatérales augmentées.

Si on peut router N^* chemins disjoints par les arêtes, le problème est polynomial (on route c unités de flot sur chacun des N^* chemins). Sinon, on peut obtenir une solution approchée de valeur $(N^* - 1)\mathrm{c}$ en procédant de la même façon (la valeur optimale étant au plus $N^*\mathrm{c}$). En outre, si $m \geq d$ et s'il n'existe que $|\mathcal{L}| - 1$ chemins disjoints par les arêtes, la valeur optimale est $(|\mathcal{L}| - 1)\mathrm{c}$ (et le problème est donc polynomial), car :

- S'il existe un i tel qu'aucune unité de flot n'est routée de s_i à s'_i, alors de toute évidence le multiflot insécable total est borné par $(|\mathcal{L}| - 1)\mathrm{c}$;
- Sinon, il existe nécessairement deux flots ϕ_r et ϕ_s qui partagent au moins une arête (puisqu'il n'existe que $|\mathcal{L}| - 1$ chemins disjoints), et on a donc $\sum_{i=1}^{|\mathcal{L}|} \phi_i = \sum_{i \neq r,s} \phi_i + (\phi_r + \phi_s) \leq (|\mathcal{L}| - 2)\mathrm{c} + \mathrm{c} \leq (|\mathcal{L}| - 1)\mathrm{c}$, où ϕ_i est le flot routé de s_i à s'_i pour tout i.

En fait, nous conjecturons le fait suivant :

Conjecture 5.1. *Opt(*MFEMI*)* $= \mathrm{c} \cdot$ *Opt(*MCDA*) dans les grilles bilatérales augmentées de capacité uniforme* c*. Par conséquent,* MFEMI *est polynomial dans ce cas.*

Remarquons que, dans le cas général, ceci n'est pas vrai. Considérons par exemple une instance de MFEMI dans une étoile ayant trois feuilles, une capacité uniforme de 2, et une liaison entre toute paire de feuilles. On a $Opt(\mathrm{MFEMI}) = 3$ (une unité de flot routée entre chacune des trois paires de feuilles) et $Opt(\mathrm{MCDA}) \times \mathrm{c} = 2$ (car $Opt(\mathrm{MCDA}) = 1$).

5.5.5 Aspects algorithmiques

Résoudre (IPL) **et** (PC)**.** Dans les sections 5.5.1 et 5.5.3, nous avons montré comment (IPL) peut être utilisé pour résoudre MCDA et MFEM respectivement. Dans la section 5.5.2, nous montrons, au moyen d'une relation de dualité, qu'une solution optimale entière pour (PC) est une solution optimale pour MCM. Dans cette section, nous décrivons deux algorithmes combinatoires, ACL et ACC, résolvant respectivement (IPL) et (PC). En outre, la preuve de validité détaillée ci-dessous montre que ACL fournit également une solution optimale pour tout $(IPL(u_1, u_2))$ (ceci étant nécessaire dans la preuve du théorème 5.7 de la section 5.5.1).

PROCÉDURE *ACL // ACL résout* (IPL)
Entrée: Une grille $(G, \mathcal{L})$ à m lignes, avec $\mathcal{L} = \{l_1, \ldots, l_{|\mathcal{L}|}\}$
Sortie: $\mathcal{L}^- \subseteq \mathcal{L}$ tel que $|\mathcal{L}^-| = N^*$ et la densité de $(G, \mathcal{L}^-)$ est m

Trier les liaisons de telle sorte que l_{i+1} soit plus D-longue que l_i, pour chaque $i \in \{1, \ldots, |\mathcal{L}| - 1\}$;
Pour chaque i **de** 1 **à** $|\mathcal{L}|$ **faire**
  Sélectionner l_i dans $\mathcal{L}^-$ si cela n'augmente aucune densité au-delà de m ;
  // si on résout $(IPL(u_1, u_2))$*, remplacer simplement* m *par* $m - 1$
  // pour chaque $v_j, j \in \{col(u_1), \ldots, col(u_2) - 1\}$
FinPour

On ne peut lancer l'exécution de ACC qu'après la fin de l'exécution de ACL, puisque $\mathcal{L}^-$ doit avoir été calculé. Pour décrire ACC, nous devons présenter la notion de **zone d'influence**. Nous verrons en (5.18) qu'une liaison sélectionnée dans ACL doit être coupée une seule fois. Par conséquent, au moment où une bande verticale v_j traversée par une liaison sélectionnée (disons l_i) est ajoutée à la multicoupe, on sait qu'aucune autre bande verticale traversée par l_i n'y sera ajoutée. Afin de garantir ceci, on définit l_j^* comme la liaison la plus G-longue traversant v_j, et la zone d'influence de v_j comme la région entre v_j et le terminal de gauche de l_j^*. Toute bande verticale située à gauche de v_j qui est traversée par l_j^* se trouve dans la zone d'influence de v_j. À chaque étape, la zone d'influence complète est définie comme étant l'union de toutes les zones d'influence déjà définies. Par conséquent, quand v_j est la bande verticale courante examinée par l'algorithme, mettre à jour la zone d'influence signifie trouver l_j^*. Par exemple, dans la figure 5.5, v_2 est dans la multicoupe et sa zone d'influence est définie par $l_1 = (s_1, s_1')$ (ou par (s_3, s_3')) : alors, aucune autre bande verticale se trouvant entre v_2 et la ligne verticale la plus à gauche (où se trouve s_1, le terminal de gauche de l_1) ne sera ajoutée à la multicoupe. Ainsi, dans cet exemple, v_2 est la seule bande verticale qui appartient à la multicoupe.

PROCÉDURE *ACC // ACC résout (PC)*
Entrée: Une grille $(G, \mathcal{L}^-)$ avec $G = (S, A)$
Sortie: Une multicoupe $\mathrm{C} \subseteq A$ telle que $|\mathrm{C}| = N^*$

zone d'influence := $\emptyset$;
C := $\emptyset$; // *initialement, la coupe est vide*
1. **Pour chaque** bande verticale v_j **de** droite **à** gauche **faire**
    Si v_j est saturée dans $(G, \mathcal{L}^-)$ **alors**
        Si v_j n'est pas dans la zone d'influence **alors**
            C := C $\cup$ {arêtes de v_j} ; // *v_j est ajoutée à la multicoupe*
            Mettre à jour la zone d'influence ;
        FinSi
    FinSi
FinPour
2. **Pour chaque** liaison l_i sélectionnée dans ACL qui n'est pas encore coupée **faire**
    C := C $\cup$ $\{a_i\}$; // *l_i est coupée à sa source*
FinPour

Validité. Nous prouvons à présent l'optimalité de ACL et de ACC en utilisant les conditions des écarts complémentaires liées aux programmes linéaires duaux (PL) et (PC). Rappelons que M (la matrice des contraintes de ces deux PL) est totalement unimodulaire et notons x^* et (w^*, y^*) des solutions optimales entières de (PL) et (PC) respectivement. Alors, les conditions des écarts complémentaires sont données par :

– D'après (5.9) : $y_j^*(m - \sum_{i/l_i \text{ traverse } v_j} x_i^*) = 0$. Cela signifie :

$$v_j \text{ est dans la multicoupe } (y_j^* = 1) \text{ seulement si elle est saturée } \left(\sum_{i/l_i \text{ traverse } v_j} x_i^* = m \right) \tag{5.16}$$

– D'après (5.11) : $w_i^*(1 - x_i^*) = 0$. Cela signifie :

$$\text{une liaison enlevée } (x_i^* = 0) \text{ ne peut pas être coupée à sa source } (w_i^* = 0) \tag{5.17}$$

– D'après (5.13) : $x_i^*((w_i^* + \sum_{j/l_i \text{ traverse } v_j} y_j^*) - 1) = 0$. Cela signifie :

$$\text{une liaison sélectionnée } (x_i^* = 1) \text{ est coupée une fois seulement } \left(w_i^* + \sum_{j/l_i \text{ traverse } v_j} y_j^* = 1 \right) \tag{5.18}$$

Tout d'abord, notons que la solution donnée par ACL est admissible. Ensuite, les solutions données par ACL et ACC vérifient les conditions des écarts complémentaires. En effet, d'une part nous ne sélectionnons dans la multicoupe que des bandes verticales qui sont saturées (5.16), d'autre part

une liaison non sélectionnée dans ACL n'est jamais coupée à sa source (5.17), et, enfin, toute liaison sélectionnée dans ACL est coupée une fois dans ACC (au moins à l'étape 2), mais jamais deux fois ou plus (c.-à-d., on n'a jamais $w_i^* + \sum_{j/l_i \text{ traverse } v_j} y_j^* \geq 2$), à cause de la notion de zone d'influence (5.18). On doit encore montrer que la solution donnée par ACC définit bien une multicoupe, c.-à-d., que chaque liaison enlevée est effectivement coupée.

Lemme 5.10. *Toute liaison enlevée dans ACL est coupée dans ACC.*

Démonstration. Supposons qu'il existe une liaison enlevée qui n'est pas coupée, disons l. Soit v la bande verticale la plus à gauche qui soit traversée par l et qui soit saturée dans $(G, \mathcal{L}^-)$ (v existe, puisque sinon l aurait été sélectionnée dans ACL). Soit $\hat{v}$ la bande verticale la plus à gauche parmi celles situées à droite de v et qui sont dans la multicoupe ($\hat{v}$ existe, puisque sinon v est la bande verticale saturée la plus à droite dans $(G, \mathcal{L}^-)$, et donc v appartient à la multicoupe et l est coupée), et soit $\hat{l}$ la liaison définissant la zone d'influence de $\hat{v}$. v est dans cette zone d'influence, puisque sinon v est dans la multicoupe et donc l est coupée.

l a été examinée avant $\hat{l}$, puisque $\hat{l}$ est strictement plus D-longue que l (car $\hat{l}$ traverse $\hat{v}$, et pas l). En outre, d'après le choix de v, toutes les bandes verticales saturées traversées par l sont traversées par $\hat{l}$. Donc l aurait du être sélectionnée à la place de $\hat{l}$ dans ACL ; une contradiction. Ceci conclut la preuve. □

Implémentation efficace. Nous prouvons à présent que ACL et ACC peuvent s'exécuter en temps $O(|\mathcal{L}|)$, et donc que tous deux ont des temps d'exécution asymptotiquement optimaux. D'après le théorème 5.1, on ne modifie pas la solubilité de l'instance si on suppose qu'il n'y a pas plus de deux colonnes consécutives sans terminaux dans la grille. Puisqu'il y a $2|\mathcal{L}|$ terminaux, on peut supposer sans perte de généralité que $n \leq 6|\mathcal{L}|$.

En outre, comme $(G, \mathcal{L})$ est une grille de capacité uniforme c, donner c, m, n et les $|\mathcal{L}|$ paires $(col(s_i), col(s'_i))$ (ainsi que l'indication, pour chaque terminal, du bord horizontal auquel il appartient) est suffisant pour décrire entièrement les données. Ainsi, nous supposons que l'entrée de ACL est un tableau $\mathcal{T}_{ACL}$ de taille $n = \Theta(|\mathcal{L}|)$, où le j^e élément contient, pour chacun des deux (ou moins) terminaux se trouvant sur la j^e colonne de la grille, la colonne du terminal associé (c'est-à-dire du puits associé si c'est la source d'une liaison, de la source associée sinon). Si, par exemple, un ensemble de paires $(col(s_i), col(s'_i))$ est donné, on peut facilement calculer $\mathcal{T}_{ACL}$ en $O(|\mathcal{L}|)$, en parcourant cet ensemble de paires une fois. Exécuter ACL peut être fait en résolvant une instance du problème d'admission d'appels sur une chaîne (voir [1] pour les détails) : chaque bande verticale de la grille v_j devient une arête b_j de la chaîne, chaque liaison devient un *appel*, et la capacité de chaque b_j est m si on résout (IPL), et $m-1$ (si $j \in \{col(u_1), \ldots, col(u_2)-1\}$) ou m (sinon)

si on résout $(IPL(u_1, u_2))$ pour une paire (u_1, u_2) donnée. Il est montré dans [1] que ce problème peut être résolu en $O(p+q)$, où p est le nombre d'appels et q le nombre d'arêtes. Dans notre cas, on a $p = |\mathcal{L}|$ et $q = n-1 = O(|\mathcal{L}|)$, donc ACL s'exécute en $O(|\mathcal{L}|)$. Montrons que c'est également le cas pour ACC. La difficulté principale pour ACC est de mettre à jour efficacement la zone d'influence. L'entrée de ACC est le tableau $\mathcal{T}' = \mathcal{T}_{ACL} \setminus \{\text{liaisons enlevées}\}$. On peut calculer en $O(|\mathcal{L}|)$ les densités de cette nouvelle grille en utilisant (5.2) ; on les stocke ensuite dans un nouveau tableau $\mathcal{T}_\Delta$. À chaque étape, $\mathcal{T}_\Delta$ sera utilisé pour savoir si la bande verticale courante est saturée ou pas. Si cette bande est ajoutée à la multicoupe, $\mathcal{T}'$ sera alors utilisé pour trouver la liaison qui définit sa zone d'influence.

Soient $\bar{v}_j$ et $\bar{v}_{j+1}$ deux bandes verticales consécutives de la multicoupe. La liaison définissant la zone d'influence de $\bar{v}_j$ ne peut pas avoir son terminal de droite à la droite de $\bar{v}_{j+1}$, puisque sinon cette liaison traverse à la fois $\bar{v}_j$ et $\bar{v}_{j+1}$, et est donc strictement plus G-longue que la liaison définissant la zone d'influence de $\bar{v}_{j+1}$: une contradiction. Ainsi, quand une bande verticale est ajoutée à la multicoupe, trouver sa zone d'influence exige seulement d'examiner les liaisons dont les terminaux de droite sont situés entre cette bande verticale et la précédente bande appartenant à la multicoupe. Donc, pendant toute l'exécution de ACC, chaque colonne dans $\mathcal{T}'$ est examinée seulement une fois. Puisque la même chose est vraie pour chaque bande verticale, le temps d'exécution de l'étape 1 est $O(\max(|\mathcal{T}'|, |\mathcal{T}_\Delta|)) = O(|\mathcal{L}|)$. En outre, au moment où une liaison est examinée, on peut savoir si elle traversera une bande verticale appartenant à la multicoupe, et l'enlever de $\mathcal{T}'$ si c'est le cas. L'étape 2 consiste alors à couper à sa source toute liaison qui reste dans $\mathcal{T}'$. ACC s'exécute donc effectivement en temps $O(|\mathcal{L}|)$.

5.5.6 Preuve du lemme 5.7

Nous détaillons à présent la preuve du lemme 5.7, énoncé et utilisé en section 5.5.2.

Lemme 5.7. *Soit $(G, \mathcal{L})$ (avec $G = (S, A)$) une grille bilatérale non augmentée où*
(a) *toutes les arêtes horizontales ont la même capacité U_h,*
(b) *toutes les arêtes verticales ont la même capacité $U_v \geq \max_{l \in \mathcal{L}} D(l)$, sauf celles qui sont sur la colonne la plus à droite ou sur la colonne la plus à gauche, dont les capacités sont au plus U_v,*
(c) *toute bande horizontale h_j vérifie $\sum_{a \text{ est une arête de } h_j} U(a) \geq \sum_{i=1}^{|\mathcal{L}|} D(l_i)$,*
(d) *soit $U_h = U_v$, soit chaque arête verticale de la colonne la plus à gauche et de la colonne la plus à droite est aussi pondérée par U_v.*

Alors, la condition de coupe est vérifiée par tout $X \subseteq S$ si et seulement si elle est vérifiée par tout $X \subseteq S$ tel que $\delta_G(X)$ est une bande verticale de la grille.

Démonstration. La nécessité de la condition de coupe sur chaque bande verticale étant évidente, nous montrons sa suffisance. Soit $X \subseteq S$ tel que $U(\delta_G(X)) < D(\delta_{\mathcal{L}}(X))$. On peut supposer s.p.d.g. que X est connexe. Montrons alors que l'on peut trouver une bande verticale qui viole la condition de coupe.

Soit $D^* = \max_{l \in \mathcal{L}} D(l)$ et soit n le nombre de colonnes de la grille. X est "encadré" par deux colonnes, c_Γ à gauche et c_Δ à droite : cela signifie que, pour chaque sommet u dans X, u est situé entre la Γ^e et la Δ^e colonne incluse (en prenant Γ aussi grand que possible et Δ aussi petit que possible). La preuve est organisée de la façon suivante : dans la première partie, nous supposons que $\Gamma > 1$ et $\Delta < n$; dans la seconde, nous considérons le cas où $\Gamma = 1$ ou $\Delta = n$.

Supposons d'abord que $\Gamma > 1$ et $\Delta < n$. Si X ne contient aucun sommet supérieur ou inférieur, $|\delta_{\mathcal{L}}(X)| = 0$: une contradiction. Si X ne contient aucun sommet supérieur (resp. inférieur), alors, X étant connexe, $\delta_G(X)$ contient *au moins* $\Delta - \Gamma + 1$ arêtes verticales (une pour chaque colonne entre c_Γ et c_Δ), tandis que $\delta_{\mathcal{L}}(X)$ contient *au plus* $\Delta - \Gamma + 1$ arêtes, puisqu'il y a au plus $\Delta - \Gamma + 1$ terminaux entre c_Γ et c_Δ. Donc, d'après (b) :

$$U(\delta_G(X)) \geq (\Delta - \Gamma + 1)U_v \geq (\Delta - \Gamma + 1)D^* \geq D(\delta_{\mathcal{L}}(X)) \tag{5.19}$$

une contradiction. Par conséquent, X contient au moins un sommet supérieur et un sommet inférieur : X "touche" à la fois la ligne horizontale du bas et celle du haut. Étant donné un sous-ensemble $Y \subseteq S$, soit $\delta_G^v(Y)$ (resp. $\delta_G^h(Y)$) l'ensemble des arêtes verticales (resp. horizontales) dans $\delta_G(Y)$. On a $\delta_G(Y) = \delta_G^v(Y) \cup \delta_G^h(Y)$ et $U(\delta_G(Y)) = U(\delta_G^v(Y)) + U(\delta_G^h(Y))$. On transforme X en un sous-ensemble $X' \subseteq S$ contenant tous les sommets entre c_Γ et c_Δ (c.-à-d., X' est le plus petit rectangle contenant X, voir la figure 5.6(a)). Puisque X est connexe et touche la ligne horizontale du bas et celle du haut, $|\delta_G^h(X)| \geq 2m$, et on a donc :

$$U(\delta_G^h(X)) \geq 2mU_h = U(\delta_G^h(X')) \tag{5.20}$$

Par souci de simplicité, on notera $\delta_{\mathcal{L}}(X - X')$ l'ensemble des liaisons *étant dans* $\delta_{\mathcal{L}}(X)$ *et pas dans* $\delta_{\mathcal{L}}(X')$. De façon évidente, on a $D(\delta_{\mathcal{L}}(X - X')) \geq D(\delta_{\mathcal{L}}(X)) - D(\delta_{\mathcal{L}}(X'))$. Pour chaque arête verticale a qui a été enlevée de $\delta_G^v(X)$ quand X a été transformé en X', au plus une liaison a été enlevée de $\delta_{\mathcal{L}}(X)$ (une liaison ayant un terminal sur la même colonne que a), donc

$$|\delta_{\mathcal{L}}(X - X')| \leq |\delta_G^v(X)| - |\delta_G^v(X')| = |\delta_G^v(X)| \tag{5.21}$$

puisque $|\delta_G^v(X')| = 0$. Donc, on a :

$$\begin{aligned} D(\delta_{\mathcal{L}}(X)) - D(\delta_{\mathcal{L}}(X')) & \leq & D(\delta_{\mathcal{L}}(X - X')) \\ & \leq & D^*|\delta_{\mathcal{L}}(X - X')| \\ & \underbrace{\leq}_{d'après\ (b)} & U_v|\delta_{\mathcal{L}}(X - X')| \\ & \underbrace{\leq}_{d'après\ (5.21)} & U_v|\delta_G^v(X)| = U(\delta_G^v(X)) \quad (5.22) \end{aligned}$$

En combinant (5.20), (5.22) et $D(\delta_{\mathcal{L}}(X)) > U(\delta_G(X))$, on obtient :

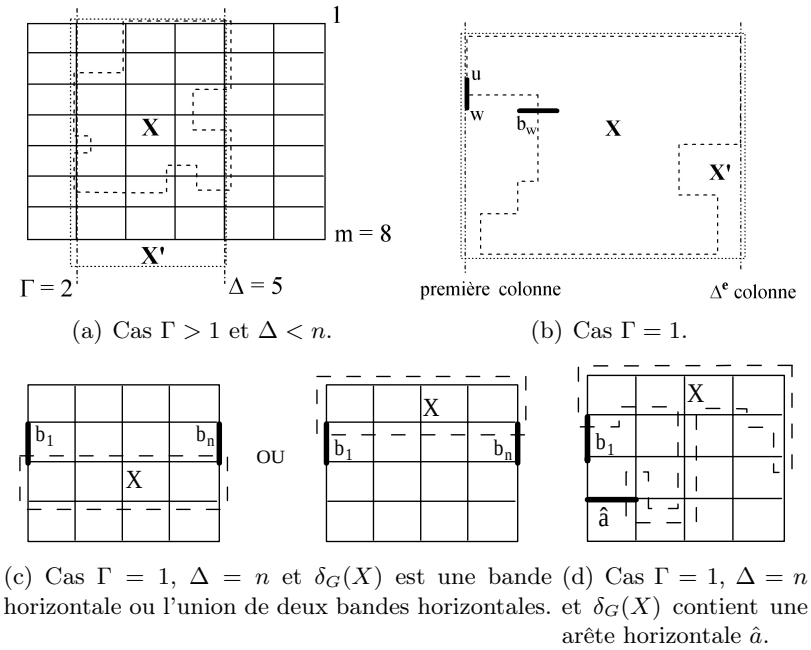
$$\begin{aligned} D(\delta_{\mathcal{L}}(X')) & \geq & D(\delta_{\mathcal{L}}(X)) - U(\delta_G^v(X)) \\ & > & U(\delta_G(X)) - U(\delta_G^v(X)) = U(\delta_G^h(X)) \\ \Rightarrow D(\delta_{\mathcal{L}}(X')) & \underbrace{>}_{d'après\ (5.20)} & U(\delta_G^h(X')) = U(\delta_G(X')) \quad (5.23) \end{aligned}$$

Par conséquent, X' viole également la condition de coupe. (5.23) implique

$$D(\delta_{\mathcal{L}}(X')) = \sum_{l_i \text{ traversant } v_{\Gamma-1}} D(l_i) + \sum_{l_i \text{ traversant } v_{\Delta}} D(l_i) > U(\delta_G(X')) = 2mU_h$$

$$\Rightarrow \text{ soit } \sum_{l_i \text{ traversant } v_{\Gamma-1}} D(l_i) > mU_h \text{ soit } \sum_{l_i \text{ traversant } v_{\Delta}} D(l_i) > mU_h$$

Ainsi, $v_{\Gamma-1}$ ou v_{Δ} viole la condition de coupe, et le lemme 5.7 est démontré. Étudions maintenant le cas où X touche la colonne la plus à gauche et/ou la plus à droite. Supposons que $\Gamma = 1$ et $\Delta < n$ (le cas où $\Delta = n$ et $\Gamma > 1$ peut être traité de manière symétrique). Si X ne touche ni la ligne du bas ni la ligne du haut, (d) implique que $\delta_G(X)$ contient au moins Δ arêtes pondérées par U_v, puisqu'il contient au moins une arête horizontale appartenant à v_{Δ}, a au moins Δ arêtes verticales (au moins une pour chaque colonne) et, d'après (b), au plus une colonne a des arêtes ayant des capacités inférieures à U_v. Ceci implique que la contradiction (5.19) reste vraie, et par conséquent X touche nécessairement la ligne inférieure et la ligne supérieure. En outre, pour chaque arête verticale $(u \in X, w \in S \setminus X)$ de la première colonne (une telle arête n'est pas dans $\delta_G(X')$), il existe une arête horizontale b_w sur la même ligne que w qui est dans $\delta_G(X)$ et pas dans $\delta_G(X')$, et qui est telle que, sur cette ligne horizontale, il n'y a aucun sommet de X entre w et le sommet le plus à gauche de b_w (puisque sinon $w \in X$, voir la figure 5.6(b)). D'après (b) et (d), soit $U((u,w)) = U_v \geq D^*$, soit $U(b_w) = U_v \geq \max(U((u,w)), D^*)$, et donc, en remplaçant "$= U(\delta_G^v(X))$" par "$\leq U(\delta_G^v(X) \cup \{b_w / \exists (u,w) \in \delta_G(X)\})$", (5.22) reste vraie dans ce cas. Enfin, $U(\delta_G^h(X) \setminus \{b_w / \exists (u,w) \in \delta_G(X)\}) \geq mU_h = U(\delta_G^h(X'))$, et donc



(a) Cas $\Gamma > 1$ et $\Delta < n$.

(b) Cas $\Gamma = 1$.

(c) Cas $\Gamma = 1$, $\Delta = n$ et $\delta_G(X)$ est une bande horizontale ou l'union de deux bandes horizontales.

(d) Cas $\Gamma = 1$, $\Delta = n$ et $\delta_G(X)$ contient une arête horizontale $\hat{a}$.

FIG. 5.6 – Exemples pour le lemme 5.7.

une preuve similaire à celle donnée ci-dessus montre que (5.23) reste vraie. $\delta_G(X')$ étant la Δ^e bande verticale, le lemme 5.7 est démontré.

Supposons maintenant que $\Gamma = 1$ et $\Delta = n$. Si X ne touche pas les lignes inférieure et supérieure, alors on a une contradiction puisqu'il existe des arêtes $b_1, \ldots, b_n$ dans $\delta_G(X)$ telles que $\sum_{p=1}^{n} U(b_p) \geq \sum_{i=1}^{|\mathcal{L}|} D(l_i)$ (avec $\sum_{i=1}^{|\mathcal{L}|} D(l_i) \geq D(\delta_{\mathcal{L}}(X))$). En effet, ou bien chaque arête verticale est pondérée par $U_v \geq D^*$ (alors, on choisit une arête verticale appartenant à $\delta_G(X)$ sur chacune des n colonnes et on obtient le résultat désiré), ou bien $\delta_G(X)$ est soit une bande horizontale soit l'union de deux bandes horizontales (alors, on applique (c), voir la figure 5.6(c)), ou bien $\delta_G(X)$ contient une arête horizontale $\hat{a}$ (voir la figure 5.6(d)), et, d'après (d), $\hat{a}$ est pondérée par U_v (ainsi, on construit un sous-ensemble E de $\delta_G(X)$ en choisissant b_n comme étant $\hat{a}$ et, pour $p \in \{1, \ldots, n-1\}$, b_p comme étant une arête verticale de la p^e colonne ; alors, d'après (b), la bande horizontale h_i contenant b_1 vérifie $(U(\delta_G(X)) \geq)$ $U(E) \geq \sum_{\text{a est une arête de } h_i} U(a)$, et on peut appliquer (c)).

Par conséquent, X touche les lignes inférieure et supérieure. Alors, pour les mêmes raisons que dans le cas où $\Gamma = 1$ et $\Delta < n$, (5.22) reste vrai, et puisque $U(\delta_G^h(X')) = 0$, on peut prouver comme précédemment que (5.23) reste vrai. Mais comme $|\delta_G(X')| = |\delta_{\mathcal{L}}(X')| = 0$, on obtient une contradiction. $\square$

5.5.7 Preuve du théorème 5.8

Rappelons que $N^* = Opt(PL)$ si $m < d$, et $N^* = |\mathcal{L}|$ sinon. Nous donnons à présent la preuve du théorème 5.8, énoncé en section 5.5.3.

Théorème 5.8. *Si* (5.4) *n'est pas vérifié, si* c *est impair, si* $|\mathcal{L}| = n$ *et si* $d \leq m < \lceil \frac{dc}{c-1} \rceil$, *alors* $Opt(\text{MFEM}) = |\mathcal{L}|\text{c} - 1$ *; Sinon* $Opt(\text{MFEM}) = N^*\text{c}$.

Démonstration. Il faut distinguer plusieurs cas. La preuve du théorème 5.8 sera une conséquence directe des preuves des quatre lemmes suivants. Dans la suite, comme en section 5.5.2, à chaque fois que $m \geq d$, on ne fera pas la distinction entre $(G, \mathcal{L})$ et $(G, \mathcal{L}^-)$. Rappelons aussi que, lorsque l'on considère une instance du problème de multiflot avec demandes, on peut transformer cette instance en une instance équivalente dans une grille non augmentée (voir la section 5.5.2). Donc, lorsqu'on considère ce problème, on suppose toujours que les grilles sont non augmentées. Le tableau 5.1 résume les résultats des lemmes 5.11, 5.12, 5.13 et 5.14.

<table>
<tr><td colspan="6">$\text{c} \geq 2$</td></tr>
<tr><td>c est pair</td><td colspan="5">c est impair</td></tr>
<tr><td rowspan="3">Lemme 5.11
Val. opt.
$= N^*\text{c}$</td><td>$m < d$</td><td colspan="2">$d \leq m < \lceil \frac{dc}{c-1} \rceil$</td><td colspan="2">$m \geq \lceil \frac{dc}{c-1} \rceil$</td></tr>
<tr><td rowspan="2">Lemme 5.12
Val. opt.
$= N^*\text{c}$</td><td>$|\mathcal{L}| < n$ ou (5.4) est vérifié</td><td>$|\mathcal{L}| = n$ et (5.4) n'est pas vérifié</td><td>$|\mathcal{L}| < n$ ou (5.4) est vérifié</td><td>$|\mathcal{L}| = n$ et (5.4) n'est pas vérifié</td></tr>
<tr><td>Lemme 5.12
Val. opt.
$= |\mathcal{L}|\text{c}$</td><td>Lemme 5.14
Val. opt.
$= |\mathcal{L}|\text{c} - 1$</td><td>Lemme 5.12
Val. opt.
$= |\mathcal{L}|\text{c}$</td><td>Lemme 5.13
Val. opt.
$= |\mathcal{L}|\text{c}$</td></tr>
</table>

TAB. 5.1 – Résumé de la preuve du théorème 5.8.

Le cas où c est pair est immédiat.

Lemme 5.11. *Supposons que* c *soit pair. Alors* $Opt(\text{MFEM}) = N^*\text{c}$.

Démonstration. Comme dans la preuve du lemme 5.9, on considère $(G, \mathcal{L}^-)$ et on définit une instance $\mathcal{I} = (G, \mathcal{L}^-, U, D)$ du problème de multiflot avec demandes, telle que $D(l_i) = \text{c}$ pour chaque liaison l_i dans $\mathcal{L}^-$. Rappelons que la fonction de capacité est donnée par $U(a) = \text{c}$, pour chaque arête a. Comme c est entier, U et D sont à valeurs entières. En outre, $U(a)$ et $D(l_i)$ étant pairs pour chaque arête a et chaque liaison l_i respectivement, la condition eulérienne (5.7) est vérifiée. Puisque $(G, \mathcal{L}^-)$ vérifie $m \geq d$, l'inégalité $\sum_{l_i \text{ traversant } v_j} D(l_i) \leq d\text{c} \leq m\text{c} = \sum_{a \text{ est une arête horizontale de } v_j} U(a)$ est vérifiée pour chaque j, et ainsi la condition de coupe est vérifiée sur chaque bande verticale. De surcroît, $\mathcal{I}$ vérifie (a), (b), (c) et (d) (voir la section 5.5.6) ; ainsi, d'après le lemme 5.7, le théorème 5.2 s'applique. D'après le corollaire 5.3, ceci nous fournit un multiflot entier ayant la même valeur qu'une

multicoupe, et qui est donc optimal : le lemme 5.11 est démontré. La preuve du théorème 5.2 étant constructive, cela fournit également un algorithme pour router les flots entiers. □

Dans la suite de la section 5.5.7, nous supposons que c est impair. Le lemme 5.12 règle plusieurs cas.

Lemme 5.12. *Supposons que* c *soit impair. Supposons en outre que* $m < d$*, ou que* $m \geq d$ *et soit* $N^* = |\mathcal{L}| < n$ *soit* (5.4) *est vérifié. Alors, la valeur optimale de MFEM est* N^*c.

Démonstration. Si $m < d$ et si m est impair alors, d'après le corollaire 5.2, $Opt(\text{MCDA}) = N^*$. Si $m < d$, si m est pair et si $(G, \mathcal{L}^-)$ vérifie (5.4), (5.5) ou (5.6), alors, d'après le théorème 5.1, on a aussi $Opt(\text{MCDA}) = N^*$. Il en est de même lorsque soit $m \geq d$ et (5.4) est vérifié, soit $m > d$ et $|\mathcal{L}| < n$. En outre, si $m = d$ et m est impair alors, d'après la proposition 5.3, $(G, \mathcal{L})$ vérifie (5.5) et, par conséquent, $Opt(\text{MCDA}) = |\mathcal{L}| = N^*$. Notons que c'est également vrai lorsque $m = d$, que m est pair et que $(G, \mathcal{L})$ vérifie soit (5.5) soit (5.6). Dans tous ces cas, nous prouvons le lemme 5.12 en utilisant le fait que, pour MFEM, une solution admissible de valeur N^*c est obtenue en routant c unités de flot sur chacun des N^* chemins disjoints par les arêtes. Comme dans la preuve du lemme 5.11, le corollaire 5.3 implique que ceci fournit un multiflot entier ayant la même valeur qu'une multicoupe, et qui est donc optimal. Dans tous ces cas, cela montre que, à chaque fois que $(G, \mathcal{L}^-)$ vérifie les hypothèses du théorème 5.1, résoudre MFEM se fait en résolvant MCDA.

Le dernier cas à considérer dans ce lemme est le cas où c est impair, $m \leq d$, $N^* < n$, m est pair et $(G, \mathcal{L}^-)$ ne vérifie aucune des trois conditions (5.4), (5.5) et (5.6).

Pour traiter ce cas, on doit seulement montrer que l'instance du problème de multiflot avec demandes $\mathcal{I} = (G, \mathcal{L}^-, U, D)$, avec $U(a) = \text{c}$ pour chaque arête a et $D(l_i) = \text{c}$ pour chaque liaison l_i dans $\mathcal{L}^-$, admet une solution entière. Cela fournira un multiflot entier de valeur $|\mathcal{L}^-|\text{c} = N^*\text{c}$. Nous commençons par transformer l'instance en une nouvelle instance en diminuant la capacité de plusieurs arêtes et en ajoutant des liaisons "virtuelles" à $(G, \mathcal{L}^-)$, de sorte que la nouvelle instance vérifie la condition eulérienne (5.7), et nous appliquons ensuite le théorème 5.2. De façon évidente, si la nouvelle instance admet une solution, alors l'instance initiale admet une solution entière. D'abord, pour $(G, \mathcal{L}^-)$, on a :

Assertion 5.1. *Soit* h *le nombre de bandes verticales saturées dans* $(G, \mathcal{L}^-)$*. Si* $(G, \mathcal{L}^-)$ *ne vérifie aucune des trois conditions* (5.4)*,* (5.5) *et* (5.6)*, alors, dans chaque ensemble* X_j*,* $j \in \{2, \ldots, h\}$*, contenant tous les sommets des bords horizontaux qui sont situés entre la* $j - 1^e$ *et la* j^e *bande verticale saturée, il y a soit zéro soit deux sommets non terminaux. En outre, si* X_1

(resp. X_{h+1}) désigne l'ensemble des sommets des bords horizontaux qui sont situés à gauche (resp. à droite) de la bande saturée la plus à gauche (resp. la plus à droite), alors tout sommet dans X_1 (resp dans X_{h+1}) est un terminal.

Démonstration. Considérons d'abord la deuxième partie de l'assertion 5.1 : d'après le théorème 5.1, X_1 (resp. X_{h+1}) est tel que tous ses sommets sont des sommets terminaux, puisque sinon $(G, \mathcal{L}^-)$ vérifie (5.5). En outre, dans chaque X_j, $j \in \{2, \ldots, h\}$, il y a au plus deux sommets non terminaux (un sur chaque bord horizontal), puisque sinon $(G, \mathcal{L}^-)$ vérifie (5.6). Pour prouver la première partie de l'assertion 5.1, supposons qu'il existe un j tel que dans X_j, il y a un seul sommet non terminal u. Soit v_i^* la i^e bande verticale saturée dans $(G, \mathcal{L}^-)$, et soit d_i^* sa densité. Notons que dans $(G, \mathcal{L}^-)$ il y a un nombre pair de sommets non terminaux sur les bords horizontaux, puisqu'une liaison a deux terminaux. On peut donc coupler ces sommets non terminaux ensemble, et former de nouvelles *liaisons virtuelles*, obtenant ainsi une grille pleine : soit $(G, \hat{\mathcal{L}}^-)$ cette nouvelle grille. u a été couplé avec un sommet w : supposons sans perte de généralité que w est situé à droite de v_j^*. Rappelons que, dans $(G, \mathcal{L}^-)$, d_{j-1}^* et d_j^* sont égales à m. En raison de la structure de $(G, \mathcal{L}^-)$, toute nouvelle liaison virtuelle traversant v_j^* (resp. v_{j-1}^*) traverse v_{j-1}^* (resp. v_j^*), sauf (u, w) qui traverse seulement v_j^*. Donc, si $\hat{N}_{j-1}$ désigne le nombre de liaisons virtuelles traversant v_{j-1}^*, les nouvelles densités de v_{j-1}^* et v_j^* dans $(G, \hat{\mathcal{L}}^-)$, désignées respectivement par $\hat{d}_{j-1}$ et $\hat{d}_j$, sont données par

$$\hat{d}_{j-1} = d_{j-1}^* + \hat{N}_{j-1} = m + \hat{N}_{j-1} \tag{5.24}$$

et

$$\hat{d}_j = d_j^* + (\hat{N}_{j-1} + 1) = m + \hat{N}_{j-1} + 1 \tag{5.25}$$

$(G, \hat{\mathcal{L}}^-)$ étant une grille pleine, toutes ses densités sont paires (voir la proposition 5.1 en section 5.2). Cependant, d'après (5.24) et (5.25), $\hat{d}_{j-1}$ et $\hat{d}_j$ sont de parité différente : une contradiction. L'assertion 5.1 est ainsi démontrée. □

On transforme maintenant la grille (non augmentée) en une nouvelle grille vérifiant la condition eulérienne. Pour chaque sommet u impair dans X_j, $j \in \{2, \ldots, h\}$, u est un sommet non terminal (puisque tous les autres sommets ont un degré pondéré égal à 4c), et, d'après l'assertion 5.1, il existe toujours un unique autre sommet non terminal sur la ligne inférieure ou supérieure qui appartient à X_j, disons w : on ajoute alors la liaison (u, w) à $(G, \mathcal{L}^-)$. On appelle *liaisons virtuelles* de telles liaisons, comme dans l'assertion 5.1. Soit $u_1, \ldots, u_m$ et $u_1^{'}, \ldots, u_m^{'}$ les sommets situés sur la colonne la plus à gauche et la colonne la plus à droite respectivement, tels que, pour chaque i, u_i et $u_i^{'}$ sont sur la i^e ligne horizontale. Pour chaque $i \in \{1, \ldots, \frac{m}{2}\}$, on diminue d'un la capacité des arêtes (u_{2i-1}, u_{2i}) et $(u_{2i-1}^{'}, u_{2i}^{'})$.

Soit $(G, \hat{\mathcal{L}}^-)$ la grille obtenue à partir de $(G, \mathcal{L}^-)$ en ajoutant les liaisons virtuelles puis en diminuant les capacités comme expliqué ci-dessus. On a $U((u_{2i-1}, u_{2i})) = U((u'_{2i-1}, u'_{2i})) = \mathrm{c} - 1$ pour chaque i, et $U(a) = \mathrm{c}$ pour toute autre arête a. On définit $D((u, w)) = 1$ pour chaque liaison virtuelle (u, w), et $D(l_i) = \mathrm{c}$ pour chaque liaison l_i dans $\mathcal{L}^-$. Soit $\hat{\mathcal{I}} = (G, \hat{\mathcal{L}}^-, U, D)$. U et D sont à valeurs entières, et $(G, \hat{\mathcal{L}}^-, U, D)$ vérifie la condition eulérienne, puisque, pour chaque $v \in V$, $U(\delta_G(v)) + D(\delta_{\hat{\mathcal{L}}^-}(v)) \in \{3\mathrm{c}-1, 3\mathrm{c}+1, 4\mathrm{c}\}$. En outre, une liaison virtuelle ne traverse aucune bande verticale saturée, donc $(G, \hat{\mathcal{L}}^-)$ vérifie $m \geq d$, puisque $(G, \mathcal{L}^-)$ le vérifie. Il est facile de voir que $\hat{\mathcal{I}}$ vérifie (a), (b) et (d). Finalement, nous montrons qu'elle vérifie aussi (c). D'une part, pour chaque bande horizontale h_j, $\sum_{a \text{ est une arête de } h_j} U(a) \geq n\mathrm{c} - 2$. D'autre part, on a :

$$\begin{aligned}
\sum_{l_i \in \hat{\mathcal{L}}^-} D(l_i) &= \sum_{l_i \in \mathcal{L}^-} D(l_i) + \sum_{(u,w) \text{ est une liaison virtuelle}} D((u, w)) \\
&= N^*\mathrm{c} + |\{\text{liaisons virtuelles}\}| \\
&= N^*\mathrm{c} + (n - N^*) = N^*(\mathrm{c} - 1) + n
\end{aligned}$$

Puisque $N^* \leq n - 1$, $N^*(\mathrm{c} - 1) + n \leq n\mathrm{c} + (1 - \mathrm{c})$. On a $\mathrm{c} \geq 2$ et c est impair, donc $1 - \mathrm{c} \leq -2$. Par conséquent, $\sum_{l_i \in \hat{\mathcal{L}}^-} D(l_i) \leq n\mathrm{c} - 2 \leq \sum_{a \text{ est une arête de } h_j} U(a)$. Mais alors (c) est vérifié, et donc le lemme 5.7 et le théorème 5.2 s'appliquent. Par conséquent, $\hat{\mathcal{I}}$ admet une solution entière, et on peut ainsi router un multiflot entier de valeur $N^*\mathrm{c} + |\{\text{liaisons virtuelles}\}|$ pour $\hat{\mathcal{I}}$, et donc de valeur $N^*\mathrm{c}$ pour l'instance initiale $\mathcal{I}$. Ceci termine la preuve du lemme 5.12. □

On doit encore traiter le cas où $m \geq d$, $N^* = |\mathcal{L}| = n$, c est impair et $(G, \mathcal{L})$ ne vérifie pas (5.4).

Lemme 5.13. *Si* c *est impair, si* $|\mathcal{L}| = n$ *et si* $m \geq \lceil \frac{d\mathrm{c}}{\mathrm{c}-1} \rceil$, *alors on a* $Opt(\mathrm{MFEM}) = |\mathcal{L}|\mathrm{c}$.

Démonstration. Pour chaque liaison l_i dans $(G, \mathcal{L})$, on définit $D(l_i) = \mathrm{c}$. On utilise la même idée que dans le lemme 5.12 : on transforme la grille (non augmentée) en une nouvelle grille qui vérifie la condition eulérienne. Les seuls sommets impairs sont ceux situés sur la colonne la plus à gauche et sur la colonne la plus à droite, puisque tous les autres sommets ont un degré pondéré égal à 4c. On construit la nouvelle grille en diminuant de un la capacité de chaque arête horizontale, et on a ainsi $U(a) = \mathrm{c} - 1$ pour chaque arête horizontale a et $U(a') = \mathrm{c}$ pour chaque arête verticale a'. Soit $(G, \hat{\mathcal{L}})$ cette nouvelle grille et soit $\hat{\mathcal{I}} = (G, \hat{\mathcal{L}}, U, D)$. U et D sont à valeurs entières, et $(G, \hat{\mathcal{L}}, U, D)$ vérifie (a), (b), (c) et (d), et la condition eulérienne également, puisque, pour chaque $v \in V$, $U(\delta_G(\{v\})) + D(\delta_{\hat{\mathcal{L}}}(\{v\})) \in \{3\mathrm{c} - 1, 4\mathrm{c} - 2\}$. En outre, $(G, \hat{\mathcal{L}})$ vérifie la condition de coupe sur chaque bande verticale

v_j puisque, pour chaque j, $\sum_{l_i \text{ traversant } v_j} D(l_i) \leq d\mathrm{c} \leq (\lceil \frac{d\mathrm{c}}{\mathrm{c}-1} \rceil)(\mathrm{c}-1) \leq m(c-1) = \sum_{a \text{ est une arête de } v_j} U(a)$. Par conséquent, le lemme 5.7 et le théorème 5.2 s'appliquent, et $\hat{\mathcal{I}}$ admet une solution entière : le lemme 5.13 est donc démontré. □

Le lemme 5.14 règle le dernier cas.

Lemme 5.14. *Supposons que* $(G, \mathcal{L})$ *ne vérifie pas* (5.4)*, que* c *soit impair, que* $|\mathcal{L}| = n$ *et que* $d \leq m < \lceil \frac{d\mathrm{c}}{\mathrm{c}-1} \rceil$*. Alors la valeur optimale de MFEM est* $|\mathcal{L}|\mathrm{c} - 1$.

Démonstration. Nous montrons d'abord que l'on peut router au plus $|\mathcal{L}|\mathrm{c}-1$ unités de flot. Pour ce faire, nous avons seulement à prouver que l'instance du problème de multiflot avec demandes $\mathcal{I} = (G, \mathcal{L}, U, D)$, avec $U(a) = \mathrm{c}$ pour chaque arête a et $D(l_i) = \mathrm{c}$ pour chaque liaison l_i dans $\mathcal{L}$, n'admet pas de solution entière. Supposons qu'elle en admette une. Le point principal de notre preuve est que seule une quantité paire d'unités de flot peut traverser chaque arête horizontale. En effet, le volume total de flot à router est exactement égal à la capacité de chaque bande horizontale, et, par conséquent, pour chaque arête horizontale a, pour chaque unité de flot routée de "droite à gauche" à travers a, il doit y avoir une unité de flot routée "de gauche à droite" à travers a. Donc, étant donné un routage admissible, la capacité inutilisée sur chaque arête horizontale est au moins 1, et on peut ainsi ramener chaque capacité horizontale de c à $\mathrm{c} - 1$ sans affecter le routage. Mais alors, $m(\mathrm{c}-1) < d\mathrm{c}$ (puisque $m < \lceil \frac{d\mathrm{c}}{\mathrm{c}-1} \rceil$) et la condition de coupe n'est plus vérifiée : une contradiction.

Nous prouvons maintenant que l'on peut router un multiflot entier de valeur $|\mathcal{L}|\mathrm{c} - 1$. On définit les sommets u_i et u'_i comme dans le lemme 5.12. Soit $l_1 = (s_1, s'_1)$ la liaison de $\mathcal{L}$ dont le terminal de gauche est u_1, c.-à-d., le coin supérieur gauche de la grille (non augmentée) (on suppose sans perte de généralité que u_1 est s_1). On enlève des lignes de $(G, \mathcal{L})$ jusqu'à ce que $m = d$. Évidemment, d (et donc m) est pair puisque $|\mathcal{L}| = n$ implique que $(G, \mathcal{L})$ est une grille pleine (voir la proposition 5.1). On diminue de un la capacité de chaque arête (u_{2i}, u_{2i+1}), $i \in \{1, \ldots, \frac{m}{2} - 1\}$, et de chaque arête (u'_{2i-1}, u'_{2i}), $i \in \{1, \ldots, \frac{m}{2}\}$. On diminue également de un la capacité de chaque arête horizontale de la m^e ligne qui est située entre la première et la $col(s'_1)^e$ colonne. On définit les demandes comme étant $D(l_i) = \mathrm{c}$ pour chaque liaison $l_i \neq l_1$ dans $\mathcal{L}$, $D(l_1) = \mathrm{c}-1$. On a $U(a) = \mathrm{c}$ pour chaque arête a, sauf pour les arêtes dont les capacités ont été diminuées à $\mathrm{c} - 1$ comme expliqué. Soit $(G, \hat{\mathcal{L}})$ la grille obtenue à partir de $(G, \mathcal{L})$ en diminuant ces capacités, et soit $\hat{\mathcal{I}} = (G, \hat{\mathcal{L}}, U, D)$.

On va montrer une variante du lemme 5.7 en adaptant la preuve de ce lemme. Nous suivons donc le schéma de sa preuve, et nous ne détaillerons que les changements notables. Nous commençons par le cas $\Gamma > 1$ et $\Delta < n$. Les

preuves de (5.19), (5.21) et (5.22) sont inchangées, par contre (5.20) doit être modifié. En effet, on n'a pas toujours $U(\delta_G^h(X)) \geq U(\delta_G^h(X'))$ (voir la figure 5.7). Par contre, on a $U(\delta_G^h(X)) \geq U(\delta_G^h(X')) - 1$ (l'égalité se produisant dans le cas illustré dans la figure 5.7), puisque $\delta_G^h(X)$ contient au plus une arête horizontale de poids $\mathrm{c} - 1$. En outre, le cas "défavorable" est celui où $s_1' \notin X$ et $s_1' \in X'$ (car, dans tous les autres cas, $U(\delta_G^h(X)) \geq U(\delta_G^h(X'))$, et le reste de la preuve est donc identique à la preuve du lemme 5.7). Dans ce cas, comme $s_1 \notin X$ (car $\Gamma > 1$), on a $(s_1, s_1') \notin \delta_{\mathcal{L}}(X)$ et il s'ensuit $|\delta_{\mathcal{L}}(X - X')| < |\delta_G^v(X)|$. On peut donc montrer (5.23) de la façon suivante : $D(\delta_{\mathcal{L}}(X')) \geq D(\delta_{\mathcal{L}}(X)) - U(\delta_G^v(X)) + 1 > U(\delta_G^h(X)) + 1 \geq U(\delta_G^h(X'))$, et on termine la preuve comme dans le lemme 5.7.

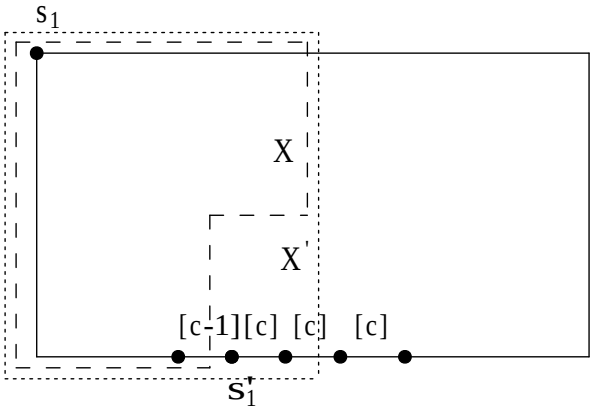


FIG. 5.7 – Un exemple du cas "défavorable".

Étudions à présent le cas où $\Gamma > 1$ et $\Delta = n$. Si X ne touche pas les lignes inférieure et supérieure, alors $\delta_G(X)$ contient Γ arêtes verticales (pondérées par $\mathrm{c} = U_v$, sauf une (au plus), pondérée par $\mathrm{c}-1$) et une arête horizontale (de poids au moins $\mathrm{c}-1$) ; on a donc $U(\delta_G(X)) \geq (\Gamma-1)\mathrm{c}+(\mathrm{c}-1)+(\mathrm{c}-1) \geq \Gamma\mathrm{c}$ (car $\mathrm{c} \geq 2 \Rightarrow 2\mathrm{c} - 2 \geq \mathrm{c}$). Ainsi, la contradiction (5.19) est toujours vraie ; X touche donc les deux bords horizontaux. Le reste de la preuve est alors identique à la preuve du lemme 5.7 (car (5.20) est vrai, c.-à-d., $U(\delta_G^h(X)) \geq U(\delta_G^h(X'))$, même dans le cas "défavorable" où $s_1' \notin X$ et $s_1' \in X'$).

Dans le cas où $\Gamma = 1$ et $\Delta < n$, on montre que X touche les deux bords horizontaux par une preuve similaire au cas précédent (en prouvant que $U(\delta_G(X)) \geq \Delta\mathrm{c}$). Ensuite, on suit le schéma de la preuve du lemme 5.7. Comme dans le cas où $\Gamma > 1$ et $\Delta < n$, la seule modification de la preuve concerne l'équation (5.20) dans le cas "défavorable" (où $s_1' \notin X$ et $s_1' \in X'$). Plaçons-nous dans ce cas. Remarquons qu'on a $U(\delta_G^h(X) \setminus \{b_w / \exists (u,w) \in \delta_G(X)\}) \geq U(\delta_G^h(X')) - 1$. Si $s_1 \in X$, alors $D(\delta_{\mathcal{L}}(X - X')) \leq (D^* - 1) + (|\delta_{\mathcal{L}}(X - X')| - 1)D^* \leq (|\delta_{\mathcal{L}}(X - X')|)D^* - 1$ (car $(s_1, s_1') \in \delta_{\mathcal{L}}(X - X')$), et

on peut donc montrer (5.23) de la façon suivante : $D(\delta_{\mathcal{L}}(X')) \geq D(\delta_{\mathcal{L}}(X)) - U(\delta_G^v(X) \cup \{b_w / \exists (u,w) \in \delta_G(X)\}) + 1 > U(\delta_G^h(X) \setminus \{b_w / \exists (u,w) \in \delta_G(X)\}) + 1 \geq U(\delta_G^h(X'))$, et on termine la preuve comme dans le lemme 5.7. Sinon (si $s_1 \notin X$), on a $(s_1, s_1') \notin \delta_{\mathcal{L}}(X)$, et donc $|\delta_{\mathcal{L}}(X - X')| < |\delta_G^v(X)|$. On peut alors montrer (5.23) de la façon suivante : $D(\delta_{\mathcal{L}}(X')) \geq D(\delta_{\mathcal{L}}(X)) - U(\delta_G^v(X) \cup \{b_w / \exists (u,w) \in \delta_G(X)\}) + 1 > U(\delta_G^h(X) \setminus \{b_w / \exists (u,w) \in \delta_G(X)\}) + 1 \geq U(\delta_G^h(X'))$, et on finit la preuve comme dans le lemme 5.7.

Le dernier cas à considérer est le cas où $\Gamma = 1$ et $\Delta = n$. X touche les deux bords horizontaux, car sinon, soit $\delta_G(X)$ contient une bande horizontale et donc $U(\delta_G(X)) \geq |\mathcal{L}|\mathrm{c} - 1 = D(\delta_{\mathcal{L}}(X))$, soit $\delta_G(X)$ contient au moins une arête horizontale (de poids au moins $\mathrm{c} - 1$) et donc $U(\delta_G(X)) \geq (|\mathcal{L}|\mathrm{c} - 2) + (c - 1) \geq |\mathcal{L}|\mathrm{c} - 1 = D(\delta_{\mathcal{L}}(X))$ (car $\mathrm{c} \geq 2$) ; dans les deux cas on obtient une contradiction. Le reste de la preuve suit le schéma de celle du lemme 5.7.

En conclusion, $(G, \hat{\mathcal{L}})$ vérifie la condition de coupe puisqu'elle la vérifie sur chacune des bandes verticales. En outre, U et D sont à valeurs entières, et la condition eulérienne (5.7) est vérifiée. Le théorème 5.2 s'applique donc, ce qui implique que $\hat{\mathcal{I}}$ admet une solution entière. Le lemme 5.14 est ainsi démontré. □

Ceci termine la preuve du théorème 5.8. □

5.5.8 Un algorithme pour router un multiflot continu de valeur $N^*\mathrm{c}$ dans une grille uniforme bilatérale augmentée

Le lemme 5.9 montre qu'il existe un multiflot continu de valeur $N^*\mathrm{c}$ dans les grilles uniformes bilatérales augmentées. Dans cette partie, nous donnons une preuve alternative de ce lemme en décrivant un algorithme simple qui fournit un multiflot continu ayant cette valeur. Il s'appuie sur l'algorithme décrit en section 5.3.4 pour résoudre le problème MCLM dans les canaux denses avec $U_h = 1$ et $U_v = 2$.

Tout d'abord, si on peut router N^* chemins disjoints par les arêtes, alors il existe un multiflot entier de valeur $N^*\mathrm{c}$; par conséquent, ce cas est trivial en utilisant les résultats de la section 5.5.1. On s'intéresse donc au cas où seuls $N^* - 1$ chemins disjoints par les arêtes peuvent être routés. Rappelons que, dans ce cas, $(G, \mathcal{L}^-)$ ne vérifie pas (5.4) (la grille est donc strictement bilatérale) ; par contre, l'assertion 5.1 s'applique (donc, si la grille n'est pas pleine, on peut la compléter en ajoutant des liaisons virtuelles qui ne modifient pas sa densité, et obtenir ainsi une grille pleine). En outre, soit $N^* < |\mathcal{L}|$ et m est pair (car si m est impair, d'après la proposition 5.3, (5.5) est vérifié et on peut donc router N^* chemins disjoints par les arêtes), soit $N^* = |\mathcal{L}|$ et $(G, \mathcal{L}^-)$ est une grille pleine, et donc de densité d paire (on peut par conséquent supposer $m = d$ sans perte de généralité). Dans les deux cas, on a un nombre pair de lignes à considérer. On obtient donc un canal dense qui vérifie les hypothèses de la section 5.3.4, excepté le fait que la capacité verticale

n'est pas paire. Cependant, on considère dans un premier temps qu'elle l'est, et on route les liaisons en utilisant l'algorithme décrit en section 5.3.4 (en considérant les lignes deux par deux).

Ensuite, pour chaque $i \in \{1, \ldots, \frac{m}{2}\}$, pour la paire formée des $2i-1^e$ et $2i^e$ lignes, on construit un routage fractionnaire de la façon suivante :

- À chaque chemin du routage initial on associe un autre chemin ayant les mêmes extrémités (les deux chemins forment ainsi un "rectangle"), comme représenté sur la figure 5.8 ;
- On route un flot de valeur $\frac{c}{2}$ sur chaque chemin (et donc sur tout segment d'un des "rectangles" circule un flot de valeur $\frac{c}{2}$).

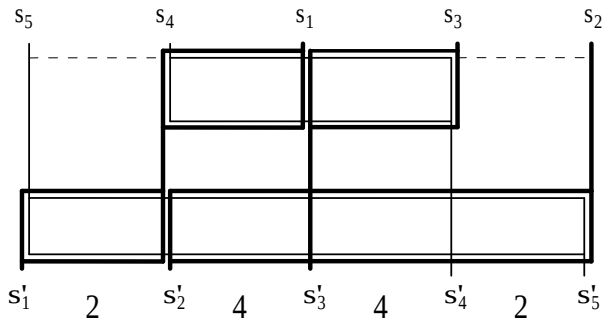


FIG. 5.8 – Routage du multiflot continu dans un canal dense.

Il n'est pas difficile de voir qu'à chaque liaison est associé un flot continu de valeur totale c et que le multiflot fractionnaire obtenu est admissible. En effet, il suffit pour cela de montrer qu'aucune contrainte de capacité n'est violée. Rappelons que, dans le routage initial, sur la paire formée des $2i-1^e$ et $2i^e$ lignes, une liaison vers la gauche et une liaison vers la droite sont routées à travers chaque bande verticale d'une région dense. Donc, d'après la façon dont sont construits les "rectangles", exactement deux d'entre eux passeront par chaque arête horizontale d'une région dense, pour un flot total de valeur $\frac{c}{2} + \frac{c}{2} = c$: la contrainte de capacité est par conséquent respectée sur toute arête horizontale. En outre, toujours dans le routage initial sur les $2i-1^e$ et $2i^e$ lignes, pour toute colonne dans une région dense $[\alpha, \beta]$ (sauf les deux colonnes α et β), le chemin d'une liaison vers la gauche (ou vers la droite) s'arrête en cette colonne et le chemin d'une nouvelle liaison vers la gauche (ou vers la droite) en est issu. En d'autres termes, d'après la façon dont sont construits les "rectangles", exactement deux d'entre eux passeront par l'arête verticale située sur cette colonne et entre les $2i-1^e$ et $2i^e$ lignes. En ce qui concerne la colonne α (resp. la colonne β), le chemin d'une liaison vers la gauche (resp. vers la droite) s'arrête en cette colonne et le chemin

d'une liaison vers la droite (resp. vers la gauche) en est issu ; le raisonnement précédent s'applique donc de façon similaire. On peut ainsi en déduire que la contrainte de capacité est respectée sur toute arête verticale, ce qui conclut la preuve de validité de l'algorithme.

Enfin, il convient de remarquer que le multiflot construit par cet algorithme est entier si c est pair : ceci fournit donc également une preuve alternative du lemme 5.11.

5.6 Conclusion

Dans ce chapitre, nous avons donné un algorithme polynomial simple pour résoudre MCLM dans des canaux denses ayant des capacités horizontale et verticale quelconques, étendant ainsi les résultats présentés dans [66], où seules des capacités unitaires sont considérées. Pour $U_h = 2$ et $U_v = 1$ et pour $U_h = 1$ et $U_v = 2$, le temps d'exécution de notre algorithme glouton est linéaire en la taille de la grille. En outre, nous avons montré que, si $U_h > 1$ ou $U_v > 1$, l'existence d'une solution implique l'existence d'une solution où toutes les liaisons sont routées suivant des plus courts chemins, alors que ceci n'est pas vrai si $U_h = U_v = 1$. Cela montre également que notre algorithme résout la variante du problème où l'on cherche à minimiser la longueur du plus long chemin.

De surcroît, nous avons prouvé que MCM et MFEMI restaient $\mathcal{NP}$-difficiles dans les grilles même sous certaines hypothèses restrictives, et nous avons exhibé un cas particulier où MCM et MFEM sont polynomiaux. L'analyse de la validité de nos algorithmes s'appuie sur des résultats antérieurs concernant des problèmes de décision connexes à nos problèmes d'optimisation [69, 70, 134], et l'algorithme combinatoire résolvant MCM s'exécute en temps linéaire. Nous avons également montré que l'écart entre les valeurs optimales de MCM et MFEM était au plus un. Bien que décider si ces deux valeurs sont égales requiert d'exécuter l'intégralité de l'algorithme, calculer une solution éloignée d'au plus une unité de l'optimum se révèle donc très rapide. Quoi qu'il en soit, nous ignorons si le facteur $O(n^2)$ dans l'énoncé du théorème 5.7 peut être amélioré. Enfin, nous voudrions faire remarquer que, dans ce chapitre, les temps d'exécution des algorithmes de routage ont, parfois, été omis à dessein. La complexité de l'algorithme résolvant MCDA et MFEM reflète donc uniquement le temps nécessaire pour décider combien d'unités de flot sont routées pour chaque liaison. Pour construire un routage explicite, il convient d'utiliser les algorithmes donnés dans [69, 70, 134, 162].

Bien qu'il serait intéressant d'étendre nos résultats de polynomialité à d'autres classes de graphes pour lesquelles on sait résoudre CDA et caractériser ses solutions (et notamment des classes de graphes où, comme les grilles, les degrés pondérés de tous les sommets ne se trouvant pas sur la face extérieure sont pairs), de tels résultats paraissent difficiles à obtenir, car, par

exemple, CDA est polynomial dans les graphes étudiés par Frank dans [70] (ces graphes sont planaires, ont tous leurs terminaux sur la face extérieure et les degrés pondérés de tous leurs sommets ne se trouvant pas sur la face extérieure sont pairs), alors que MCDA est APX-difficile dans les graphes planaires superficiels (qui sont un cas très particulier des graphes étudiés dans [70]) et MCM est APX-difficile dans les étoiles non pondérées [80].

Chapitre 6

Multicoupes dans les graphes planaires

6.1 Préliminaires

Dans ce chapitre, nous étudions la complexité de MCM dans les graphes planaires (non orientés). Nous commençons par rappeler quelques résultats utiles.

Dahlhaus et al. ont montré que, dans les graphes quelconques, CMTM (et donc MCM) était APX-difficile même avec trois terminaux (c'est-à-dire, pour MCM, avec trois liaisons) [54]. Néanmoins, si le nombre de terminaux (et donc de liaisons) est fixé, ils ont également montré que CMTM devenait polynomial dans les graphes planaires [54] (d'autres algorithmes ont été obtenus par Hartvigsen [91] et Yeh [164] ; nous détaillons celui de Yeh à la fin de cette section), et Garg et al. ont montré que MCM était polynomial dans les arbres [80]. Lorsque le nombre de liaisons et de terminaux est quelconque, Dahlhaus et al. ont montré que CMTM restait $\mathcal{NP}$-difficile dans les graphes planaires [54], alors que MCM est APX-difficile dans les étoiles non pondérées [80]. Enfin, dans les graphes planaires, CMTM est polynomial si tous les terminaux sont sur la face extérieure [42], et MCM admet un algorithme $O(1)$-approché [157] (alors que, pour le cas général, on ne connaît qu'un algorithme $O(\log|\mathcal{L}|)$-approché [79]).

Sur la base de ces résultats, l'une des questions qui nous a intéressé est la suivante : quelle est la complexité de MCM dans les graphes planaires lorsque le nombre de terminaux est fixé ? Pour tenter d'y répondre, nous avons cherché à concevoir un algorithme polynomial pour ce problème. Dans ce chapitre, nous exposons deux nouveaux résultats qui ont découlé de cette étude. Nous commençons par décrire rapidement l'une des pistes que nous avons suivies, car elle nous a poussé à nous intéresser de près à l'algorithme de Yeh pour CMTM. Ainsi, nous avons fini par découvrir que cet algorithme était en réalité incorrect, et par comprendre pourquoi : une propriété struc-

turelle fondamentale utilisée dans la preuve de cet algorithme est fausse. Nous détaillons donc un exemple où cette propriété n'est pas vérifiée. Ensuite, nous montrons comment résoudre MCM en temps polynomial dans les graphes planaires ayant un nombre de terminaux fixé et tous leurs terminaux sur la face extérieure.

Nous donnons auparavant une brève description du principe de l'algorithme de Yeh, qui a le mérite d'être plus simple que ceux de Dahlhaus et al. et d'Hartvigsen.

L'algorithme de Yeh pour CMTM. Commençons par donner quelques définitions (nous utilisons la même terminologie que dans [164]). Étant donné un graphe planaire G et une liste $\mathcal{T}$ de q terminaux $\{t_1, \ldots, t_q\}$, on note :

- G^D le dual de G ;
- C une coupe multiterminale minimum séparant G en q composantes connexes (t_i étant séparé de t_j, pour $i \neq j$) ;
- $C^D \subseteq G^D$ la coupe duale de C (c'est-à-dire l'ensemble d'arêtes dual de C);
- S_i la composante connexe contenant t_i après avoir enlevé les arêtes de C ;
- $C_i \subseteq C$ la coupe séparant t_i de $\mathcal{T} \setminus t_i$ dans C ;
- $C_i^D \subseteq C^D$ la coupe duale de C_i.

Si un sommet w de C^D est adjacent à au moins trois sommets de C^D, alors w est appelé une *jointure-duale* (*dual-joint* en anglais) dans C^D. Une jointure-duale dans G^D est une face dans G, et la frontière d'une telle face est appelée *cycle-jointure* (*joint-cycle* en anglais). H, le *graphe des composantes* de G est le graphe planaire dont les sommets α_i sont les composantes S_i de G, deux sommets α_i et α_j de H étant adjacents si et seulement si S_i et S_j sont adjacentes dans G (c'est-à-dire ssi $C_i \cap C_j \neq \emptyset$). Enfin, on note $w_{i1}, \ldots, w_{ip_i}$ les p_i jointures-duales contenues dans C_i^D, pour chaque i (c'est une numérotation arbitraire des jointures-duales : puisque chaque jointure-duale est adjacente à plusieurs C_i^D, deux appellations $w_{i_1 j_1}$ et $w_{i_2 j_2}$ peuvent très bien correspondre à la même jointure-duale). Le lemme suivant est évident :

Lemme 6.1. *Étant donné $i \in \{1, \ldots, q\}$, si S_i et C_i sont ôtés du graphe, alors $C' = C \setminus C_i$ est une coupe multiterminale minimum pour le graphe obtenu.*

En appliquant ce lemme pour i de 1 à $q-2$, on obtient :

Corollaire 6.1. *Après avoir enlevé $\bigcup_{i \in \{1,\ldots,q-2\}} S_i$ et $\bigcup_{i \in \{1,\ldots,q-2\}} C_i$, une coupe multiterminale minimum pour le graphe obtenu est simplement une coupe minimum dans ce graphe.*

L'idée de Yeh est donc de construire une coupe multiterminale en identifiant et en enlevant de façon itérative les composantes S_i. Évidemment, l'étape la plus difficile consiste à identifier ces composantes. Pour cela, Yeh énonce les propriétés suivantes :

Lemme 6.2 (Théorèmes 2 et 3 de Yeh [164]). *Pour chaque i :*

1. *C_i est une coupe qui sépare $\mathcal{A}_i = \{t_i, v^*_{i1}, \ldots, v^*_{ip_i}\}$ de $\mathcal{B}_i = \mathcal{T} \setminus \{t_i\}$, où v^*_{ij} est un sommet (dit* sommet frontalier*) se trouvant à la fois dans S_i et sur le cycle-jointure de w_{ij}, $j = 1, \ldots, p_i$;*
2. *Si $C'_i \neq C_i$ est une coupe séparant $\mathcal{A}_i$ de $\mathcal{B}_i$, alors $C' = (C \setminus C_i) \cup C'_i$ est aussi une coupe multiterminale ;*
3. *C_i est une coupe minimum séparant $\mathcal{A}_i$ de $\mathcal{B}_i$, et $(C \setminus C_i) \cap C^*_i = \emptyset$ pour toute coupe minimum C^*_i séparant $\mathcal{A}_i$ de $\mathcal{B}_i$.*

Ainsi, son algorithme consiste à choisir, pour un i, les sommets frontaliers $v^*_{i1}, \ldots, v^*_{ip_i}$, et à calculer ensuite une coupe minimum entre les ensembles $\mathcal{A}_i$ et $\mathcal{B}_i$ définis ci-dessus ; la composante obtenue est ensuite ôtée. Cette procédure doit être itérée jusqu'à ce qu'on obtienne une coupe multiterminale. Cependant, il existe de nombreuses façons de choisir ces sommets, et toutes doivent être essayées. L'argument principal de la preuve de Yeh réside dans le fait que le nombre de sommets frontaliers à choisir pour un t_i donné est au plus le nombre de cycles-jointures associés. Or, chaque cycle-jointure du graphe initial correspond à une face du graphe des composantes qui, lui, possède q sommets, et donc au plus $2q - 4$ faces (d'après (1.1) en section 1.2). Si q est borné, le nombre de sommets frontaliers à choisir est donc borné.

Yeh montre ainsi que la complexité de cet algorithme est :

$$O\left(\left(\frac{2|\mathcal{T}|-3}{2}\right)^{|\mathcal{T}|-1} \cdot (n-|\mathcal{T}|)^{2|\mathcal{T}|-4} \cdot \left(\frac{2n-3|\mathcal{T}|+1}{2}\right)|\mathcal{T}| \cdot \log(n-|\mathcal{T}|) + n|\mathcal{T}|^2(|\mathcal{T}|+\log n)\right)$$

Il est donc plus rapide que les algorithmes de Dahlhaus et al. et d'Hartvigsen. En outre, comme nous l'avons déjà mentionné, il est également beaucoup plus simple. Cependant, nous allons montrer qu'il est en réalité incorrect.

6.2 Une tentative avortée pour résoudre MCM

Nous détaillons à présent la piste qui nous a conduit à étudier de près l'algorithme de Yeh. Nous espérions résoudre MCM dans les graphes planaires ayant un nombre fixé de terminaux à l'aide d'une approche combinant une variante de la technique d'énumération des partitionnements (décrite en section 3.1) et une variante de l'algorithme de Yeh. Plus précisément, on peut remarquer que l'on ne peut pas résoudre MCM en se ramenant directement à une instance de CMTM sur un graphe planaire (qui pourrait ensuite être

résolue à l'aide de n'importe lequel des trois algorithmes décrits dans [54], [91] et [164]), car, lorsqu'on utilise la technique d'énumération des partitionnements, on doit ajouter des sommets et des arêtes additionnels, et cette opération peut détruire la planarité du graphe.

L'idée est donc la suivante : une fois que les arêtes de n'importe quelle solution optimale $\mathcal{S}$ pour une instance de MCM ont été enlevées, les sommets du graphe sont répartis dans $q(\mathcal{S}) \leq 2|\mathcal{L}|$ composantes connexes, chacune contenant au moins un terminal (tout sommet du graphe étant relié à au moins un terminal), et aucune ne contenant à la fois s_i et s'_i pour un i. Pour chaque solution optimale, on peut alors définir le partitionnement suivant pour les terminaux : pour chaque i, le i^e paquet du partitionnement contient les terminaux contenus dans la i^e composante connexe. En énumérant tous les partitionnements composés d'au plus $2|\mathcal{L}|$ paquets (ce qui nécessite d'énumérer plus de partitionnements que dans la section 3.1), on est sûr de trouver tous ceux qui sont associés à des solutions optimales pour MCM. Si l'on considère l'exemple très simple d'une chaîne non pondérée $s_1, s'_1, s_2, s'_2, s_3, s'_3, \ldots, s_{|\mathcal{L}|}, s'_{|\mathcal{L}|}$ (où $q(\mathcal{S}) = |\mathcal{L}| + 1$ pour toute solution optimale $\mathcal{S}$), le partitionnement

$$\{\{s_1\}, \{s'_1, s_2\}, \{s'_2, s_3\}, \ldots, \{s'_{|\mathcal{L}|-1}, s_{|\mathcal{L}|}\}, \{s'_{|\mathcal{L}|}\}\}$$

est associé à une solution optimale pour MCM, mais

$$\{\{s_1, s_2, \ldots, s_{|\mathcal{L}|}\}, \{s'_1, s'_2, \ldots, s'_{|\mathcal{L}|}\}\}$$

ne l'est pas (car chacun des deux paquets induit plusieurs composantes connexes).

On se retrouve ainsi à devoir résoudre le problème suivant (qui est équivalent à CMTM dans les graphes généraux, mais pas dans les graphes planaires) : étant donné un graphe dont les arêtes sont pondérées et un partitionnement des terminaux, trouver une *coupe multi-ensemble* minimum, c'est-à-dire un ensemble d'arêtes de poids minimal dont la suppression sépare $\mathcal{T}_i$ de $\mathcal{T}_j$ pour $i \neq j$, où $\mathcal{T}_i$ est le i^e ensemble de terminaux (ou *paquet*) du partitionnement. Ce problème (CMEM), introduit dans [58], est défini comme le *Problème de la Coupe Multiterminale Colorée* dans [54], où il est montré qu'il est $\mathcal{NP}$-difficile dans les graphes planaires, même si le nombre de paquets dans le partitionnement est fixé (notons que la preuve du théorème 3.4 en section 3.2.1 montre que, pour un nombre fixé de paquets dans le partitionnement, CMEM est polynomial dans les graphes de largeur d'arbre bornée). On pourrait donc penser que cette transformation est sans intérêt (puisqu'on se ramène à un problème qu'on ne sait pas résoudre) ; pourtant, il est possible de l'améliorer.

Soit $\mathcal{I}$ une instance connexe de MCM, soit $\tilde{\mathcal{S}}$ une solution optimale pour $\mathcal{I}$ telle que $q(\tilde{\mathcal{S}}) = \max_{\mathcal{S} \ / \ \mathcal{S} \text{ est une solution optimale pour } \mathcal{I}} q(\mathcal{S})$ et soit $\mathcal{I}(\tilde{\mathcal{S}})$ l'instance de CMEM définie à partir de $\mathcal{I}$ en considérant le partitionnement

associé à $\tilde{\mathcal{S}}$. Si $Opt(\mathcal{I})$ et $Opt(\mathcal{I}(\tilde{\mathcal{S}}))$ désignent respectivement les valeurs optimales de $\mathcal{I}$ et $\mathcal{I}(\tilde{\mathcal{S}})$, on a $Opt(\mathcal{I}) = Opt(\mathcal{I}(\tilde{\mathcal{S}}))$, car :

- Toute solution pour $\mathcal{I}(\tilde{\mathcal{S}})$ est une solution pour $\mathcal{I}$, donc $Opt(\mathcal{I}) \leq Opt(\mathcal{I}(\tilde{\mathcal{S}}))$;
- La valeur de $\tilde{\mathcal{S}}$ est $Opt(\mathcal{I})$ et $\tilde{\mathcal{S}}$ est admissible pour $\mathcal{I}(\tilde{\mathcal{S}})$, donc $Opt(\mathcal{I}) \geq Opt(\mathcal{I}(\tilde{\mathcal{S}}))$.

En particulier, toute solution optimale pour $\mathcal{I}(\tilde{\mathcal{S}})$ est une solution optimale pour $\mathcal{I}$. Donc, d'après la définition de $\tilde{\mathcal{S}}$, cela implique que, après avoir enlevé les arêtes de n'importe quelle solution optimale pour $\mathcal{I}(\tilde{\mathcal{S}})$, on obtient exactement $q(\tilde{\mathcal{S}})$ composantes connexes. En d'autres termes, tout paquet du partitionnement associé à $\mathcal{I}(\tilde{\mathcal{S}})$ *induit* une seule composante connexe dans toute solution optimale (c'est-à-dire que, après avoir enlevé les arêtes de la solution, tous les terminaux de ce paquet appartiennent à la même composante connexe). Il nous reste donc à résoudre $\mathcal{I}(\tilde{\mathcal{S}})$ lorsque toute solution optimale induit $q(\tilde{\mathcal{S}})$ composantes connexes.

Ce problème est un peu plus général que CMTM (et, évidemment, un peu moins que CMEM) : en effet, toute solution optimale pour une instance de CMTM avec q terminaux induit exactement q composantes connexes, donc CMTM est un cas particulier de ce problème (où chaque paquet contient un seul terminal). Néanmoins, adapter l'algorithme de Yeh à ce problème ne semble pas problématique, car il convient de noter que cet algorithme cherche à définir les *frontières* des S_i (c'est-à-dire les C_i^D) : en d'autres termes, dans ce cas, il n'y a pas de différence fondamentale entre séparer $\{t_i, v_{i1}^*, \ldots, v_{ip_i}^*\}$ de $\mathcal{T} \setminus \{t_i\}$ et séparer $\mathcal{T}_i \cup \{v_{i1}^*, \ldots, v_{ip_i}^*\}$ de $\mathcal{T} \setminus \mathcal{T}_i$ (où les v_{ij}^* sont tels que définis dans le lemme 6.2). C'est au cours de ce travail que nous nous sommes aperçu que le lemme 6.2 était faux (et, plus précisément, que les points 2 et 3 de ce lemme l'étaient).

Considérons l'instance de CMTM donnée en figure 6.1, et intéressons-nous à la coupe multiterminale optimale dessinée en pointillés. C_1^D (la frontière de S_1, avec $t_1 \in S_1$) contient quatre jointures-duales, et, sur chacun des quatre cycles-jointures associés, un seul sommet, celui qui est adjacent aux quatre arêtes en gras, appartient à S_1. La seule coupe minimum qui sépare ces quatre sommets et t_1 de $\{t_2, t_3, t_4, t_5\}$ est celle obtenue en coupant toutes les arêtes en gras ; son poids total est 24. (En comparaison, le poids de C_1^D est 28.) Si l'on remplace C_1 par cette coupe minimum, t_2, t_3, t_4 et t_5 se trouvent dans la même composante connexe alors que, d'après le lemme 6.2, on devrait obtenir une coupe multiterminale de même poids. Ceci montre donc que ce lemme est erroné ; en fait, la raison principale en est que, dans la preuve du point 2, Yeh considère que, pour tout i, le dual de C_i' est un cycle unique, alors que l'exemple donné en figure 6.1 montre qu'il peut être constitué de plusieurs cycles disjoints. Ce fait étant le principal fondement de l'algorithme et de sa preuve, on peut légitimement estimer que cet algorithme ne peut pas facilement être corrigé.

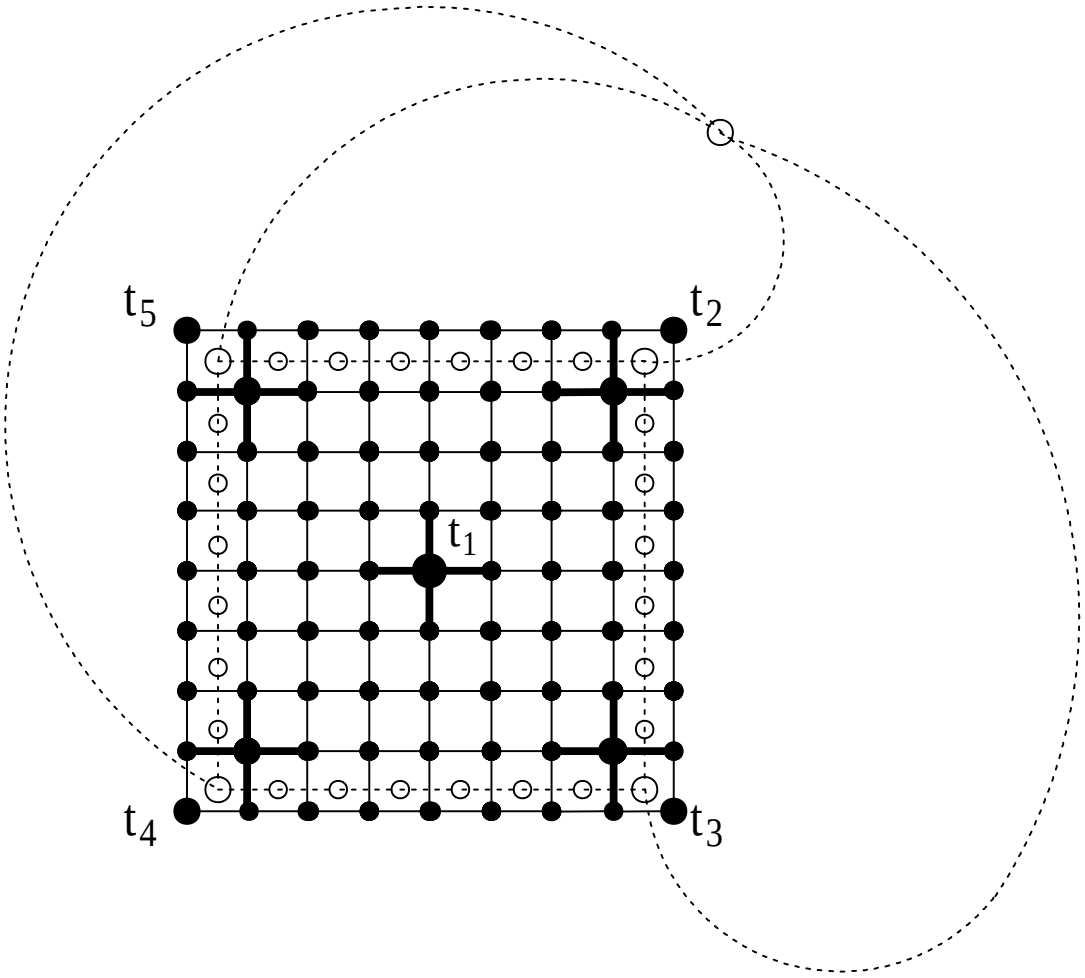


FIG. 6.1 – Un contre-exemple pour l'algorithme de Yeh. Les arêtes de la coupe multiterminale optimale dans le graphe planaire dual sont représentées en pointillés (et elles sont toutes de poids 1). Les cinq terminaux $(t_1, t_2, t_3, t_4, t_5)$ sont les gros sommets noirs, et toutes les arêtes en gras sont de poids 1, sauf les quatre arêtes en gras qui sont adjacentes à t_1 et qui sont de poids 2. Les autres arêtes du graphe ont un très grand poids (1000, par exemple).

Par ailleurs, nous avons le sentiment que l'algorithme de Dahlhaus et al. pour CMTM ne peut probablement pas être étendu à notre problème. En effet, un argument essentiel et récurrent dans leur preuve est que, si l'on considère C_i^D pour un i et si l'on remplace une partie de ce cycle (à l'intérieur duquel se trouve t_i), par exemple celle comprise entre deux sommets u et v, par une autre chaîne reliant u à v, alors t_i est soit à l'intérieur soit à l'extérieur du nouveau cycle obtenu. Dans notre cas, si l'on applique le même raisonnement, il peut y avoir une partie des terminaux de $\mathcal{T}_i$ à l'intérieur du nouveau cycle, et le reste des terminaux de $\mathcal{T}_i$ à l'extérieur de ce cycle : on ne peut donc pas utiliser cette approche de preuve par dichotomie.

Enfin, concernant l'algorithme de Hartvigsen, il partage une partie de son déroulement avec celui de Dahlhaus et al. En outre, sa preuve est basée sur des propriétés structurelles propres aux coupes multiterminales minimums et issues de la théorie des matroïdes, qui n'ont a priori aucune raison d'être vérifiées par notre problème. Nous pensons donc que l'approche d'Hartvigsen ne peut pas non plus être étendue à notre problème ; néanmoins, une étude plus approfondie serait nécessaire pour en avoir la certitude.

Malgré ces résultats négatifs, nous prouvons dans la section suivante que MCM est polynomial dans les graphes planaires ayant un nombre fixé de terminaux, si l'on suppose en plus que tous les terminaux se trouvent sur la face extérieure.

6.3 Un algorithme pour MCM dans les graphes planaires

6.3.1 Description de l'algorithme

Nous avons vu dans la section précédente que, étant donnée une instance planaire de MCM, il était impossible d'appliquer directement la technique d'énumération des partitionnements sans risquer de perdre la planarité de l'instance. Nous montrons dans cette section que, si tous les terminaux sont sur la face extérieure, alors on peut utiliser cette technique.

On suppose dans cette section que toutes les instances de MCM que l'on considère sont 2-sommet-connexes, et cette hypothèse peut être faite sans perte de généralité (et donc, dans les graphes planaires que nous considérerons, la face extérieure coïncidera avec le cycle extérieur et consistera en un cycle unique). En effet, on peut rendre n'importe quelle instance de MCM 2-arête-connexe en doublant le poids de toutes les arêtes (tous les poids sont ainsi pairs), puis en remplaçant chaque arête (u, v) (de poids $c((u, v))$) par quatre arêtes $(u, w), (w, v), (u, w')$ et (w', v) (où w et w' sont deux nouveaux sommets), de poids respectifs $\frac{c((u,v))}{2}, +\infty, \frac{c((u,v))}{2}$ et $+\infty$ (on double ainsi la valeur de toute solution, et donc, en particulier, la valeur optimale). Ensuite, on supprime tout *sommet d'articulation* (c'est-à-dire tout sommet qui appar-

tient à au moins deux blocs différents) en le remplaçant par un cycle (qu'on appellera *cycle d'articulation*) dont les arêtes sont pondérées par $+\infty$ et qui a autant de sommets que le sommet initial avait d'arêtes adjacentes. On relie alors la i^e des arêtes initialement adjacentes au sommet d'articulation au i^e sommet de ce cycle (notons que, pour que cette transformation conserve la planarité éventuelle du graphe, l'ordre des arêtes doit être celui que l'on obtient en visitant ces arêtes, dans le sens des aiguilles d'une montre par exemple, dans un plongement du graphe). S'il y avait un terminal t_i sur le sommet d'articulation, alors, dans le nouveau graphe, on place t_i sur un des sommets du cycle d'articulation associé qui se trouve sur la face extérieure (il y a deux sommets de ce type pour chaque bloc contenant t_i dans le graphe initial) ; une illustration de cette transformation est donnée en figure 6.2(a). En outre, on supposera que les sommets du cycle extérieur sont numérotés (le numéro d'un sommet v étant noté $n(v)$), à partir d'un sommet arbitraire (qui sera donc le sommet numéro 1), dans l'ordre dans lequel ils sont rencontrés en effectuant, dans le sens des aiguilles d'une montre, un parcours du cycle extérieur commençant au sommet 1. Enfin, étant donnés deux sommets u et v du cycle extérieur tels que $n(u) < n(v)$, on désignera par $p(u, v)$ les sommets de la chaîne reliant u à v sur le cycle extérieur (c'est-à-dire les sommets w du cycle extérieur tels que $n(u) \leq n(w) \leq n(v)$).

Rappelons que, étant donnée une instance de MCM, une fois que les arêtes de toute solution optimale $\mathcal{S}$ pour cette instance ont été enlevées, les sommets du graphe sont répartis dans au plus $2|\mathcal{L}|$ composantes connexes, et on définit le partitionnement associé à $\mathcal{S}$ ainsi : $\mathcal{T}_i$, le i^e paquet du partitionnement, contient les terminaux appartenant à la i^e composante connexe.

Le résultat principal dont nous aurons besoin dans cette section est le lemme suivant :

Lemme 6.3. *Étant donnée une instance de MCM dans un graphe planaire ayant ses terminaux sur la face extérieure, tout partitionnement des terminaux associé à une solution optimale pour cette instance vérifie que, pour toute paire de paquets $\mathcal{T}_i$ et $\mathcal{T}_j$ du partitionnement, pour toute paire de terminaux v_i et v'_i de $\mathcal{T}_i$ (avec $n(v_i) < n(v'_i)$) et pour toute paire de terminaux v_j et v'_j de $\mathcal{T}_j$ (avec $n(v_j) < n(v'_j)$), on a :*

$$p(v_i, v'_i) \cap p(v_j, v'_j) = \emptyset \text{ ou } p(v_i, v'_i) \subset p(v_j, v'_j) \text{ ou } p(v_j, v'_j) \subset p(v_i, v'_i) \quad (6.1)$$

Démonstration. Soit une solution optimale pour l'instance de MCM considérée : on définit v_i, v'_i, v_j et v'_j comme dans l'énoncé du lemme. D'abord, on a $v_i \neq v_j$ (par exemple), donc on ne peut pas avoir $p(v_i, v'_i) = p(v_j, v'_j)$. Ensuite, supposons que l'on ait $p(v_i, v'_i) \cap p(v_j, v'_j) \neq \emptyset$, $p(v_i, v'_i) \not\subseteq p(v_j, v'_j)$ et $p(v_j, v'_j) \not\subseteq p(v_i, v'_i)$. Alors, cela implique que les quatre sommets v_i, v'_i, v_j, v'_j vérifient $n(v_i) < n(v_j) < n(v'_i) < n(v'_j)$. Or, comme la solution considérée est optimale et comme v_i et v'_i (resp. v_j et v'_j) sont dans le même paquet (du partitionnement associé à cette solution), ils appartiennent à la même

composante connexe (une fois que les arêtes de la solution ont été enlevées). Ainsi, il existe une chaîne entre s_i et s'_i et une chaîne entre s_j et s'_j : à cause de la planarité du graphe, ces deux chaînes se "croisent" (c'est-à-dire, partagent au moins un sommet) ; voir une illustration de ce fait en figure 6.2(b). Donc, s_i, s'_i, s_j et s'_j sont en réalité dans la même composante connexe, ce qui contredit le fait que la solution considérée est valide, et termine la preuve. □

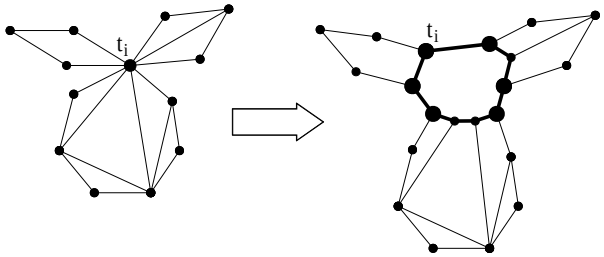
(On peut remarquer que, dans ce lemme, on pourrait remplacer "solution optimale" par "solution minimale", au sens où il n'existe pas de solution strictement incluse dans cette solution ; toute solution optimale est minimale.)

À partir de là, il n'est pas difficile de montrer que toute instance de CMEM vérifiant la propriété (6.1) peut être transformée en une instance planaire de CMTM ayant un nombre fixé de terminaux (un par paquet). Pour cela, on définit une relation d'ordre partiel sur les paquets : un paquet $\mathcal{T}_i$ est *plus court* qu'un paquet $\mathcal{T}_j$ si $p(u_i, v_i) \subset p(u_j, v_j)$, où u_i, v_i, u_j, v_j sont tels que $n(u_i) = \min_{w \in \mathcal{T}_i} n(w)$, $n(v_i) = \max_{w \in \mathcal{T}_i} n(w)$, $n(u_j) = \min_{w \in \mathcal{T}_j} n(w)$ et $n(v_j) = \max_{w \in \mathcal{T}_j} n(w)$. Pour chaque paquet du plus court au moins court (si deux paquets sont incomparables, on les considère dans un ordre quelconque), on ajoute un sommet additionnel relié (par une arête de très grand poids) à tous les terminaux de ce paquet. L'ordre considéré garantit que l'instance obtenue est effectivement planaire ; voir un exemple en figure 6.2(c). On peut cependant remarquer que, dans l'instance de CMTM obtenue, *tous les terminaux ne sont pas nécessairement sur la face extérieure* : on ne peut donc pas utiliser l'algorithme décrit dans [42]. On peut également noter que la propriété (6.1) est nécessaire pour que le partitionnement considéré soit associé à une solution optimale pour MCM, mais elle n'est pas suffisante (un contre-exemple est donné en figure 6.2(d)).

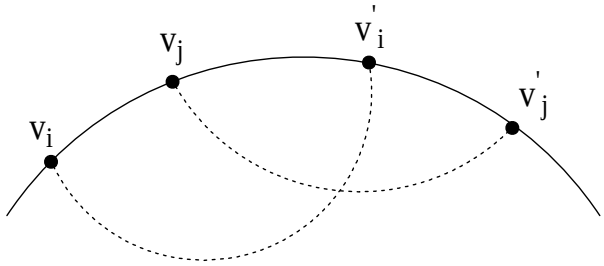
En résumé, il nous suffit d'énumérer tous les partitionnements contenant au plus $2|\mathcal{L}|$ paquets, et, pour chacun d'entre eux, de tester si la propriété (6.1) est vérifiée (car, sinon, on sait d'après le lemme 6.3 que le partitionnement ne peut pas être associé à une solution optimale) ; si c'est le cas, on calcule la meilleure solution associée en réduisant l'instance de MCM à une instance planaire de CMTM.

L'algorithme peut donc être décrit ainsi :

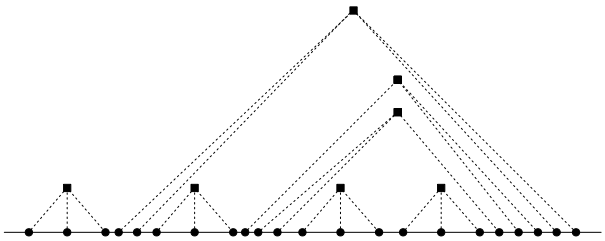
- Pour chaque partitionnement contenant au plus $2|\mathcal{L}|$ paquets :
  - Si le partitionnement courant ne vérifie pas la propriété (6.1), passer au suivant ;
  - Sinon, réduire l'instance à une instance planaire de CMTM, et résoudre l'instance obtenue à l'aide d'un des algorithmes pour CMTM (celui de [54] ou celui de [91]). Garder la solution obtenue si elle est meilleure que la meilleure solution calculée jusqu'à présent ;
- À la fin de l'algorithme, on renvoie la meilleure solution obtenue.



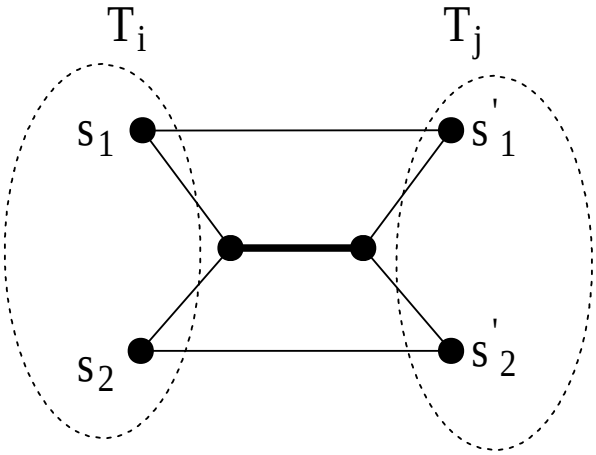
(a) Transformation d'un sommet d'articulation (les gros sommets noirs sont ceux sur la face extérieure).



(b) Un croisement inévitable si $n(v_i) < n(v_j) < n(v'_i) < n(v'_j)$.



(c) Conservation de la planarité après ajout des sommets additionnels (les carrés noirs) et des arêtes additionnelles (en pointillés).



(d) Non suffisance de la propriété (6.1). L'arête en gras est pondérée par $+\infty$.

FIG. 6.2 – Illustration de la preuve du théorème 6.1.

Le temps d'exécution de la première phase (où l'on teste la propriété (6.1)) étant négligeable par rapport à celui de la résolution de l'instance de CMTM (voir la section suivante), la complexité totale de l'algorithme est majorée par le nombre de partitionnements possibles (qui est borné par $2|\mathcal{L}|^{2|\mathcal{L}|}$) multiplié par la complexité de l'algorithme utilisé pour résoudre CMTM (qui est polynomiale d'après [54, 91]). Cet algorithme est donc bien polynomial, ce qui implique :

Théorème 6.1. *MCM est polynomial dans les graphes planaires ayant un nombre borné de terminaux, si tous les terminaux sont sur la face extérieure.*

On peut noter, en particulier, que cela inclut les cactus ayant un nombre borné de terminaux (ce cas étant également couvert par le théorème 3.4 en section 3.2.1). Par ailleurs, l'approche utilisée montre que CMEM est polynomial dans les graphes planaires ayant leurs terminaux sur la face extérieure et en nombre fixé (c.-à-d., si on fixe la somme des tailles des paquets).

6.3.2 Implémentation

Dans cette section, nous examinons quelques détails d'implémentation pour l'algorithme décrit dans la section précédente.

La première remarque à faire est que la phase où l'on teste si un partitionnement vérifie la propriété (6.1) peut être exécutée en temps $O(|\mathcal{L}|^4)$ (et son temps d'exécution est donc négligeable par rapport à la complexité de l'algorithme d'Hartvigsen, meilleur que celui de Dahlhaus et al., qui est de l'ordre de $|\mathcal{L}|4^{|\mathcal{L}|}n^{2|\mathcal{L}|-3}\log n$). En effet, on doit considérer tout couple de paquets $\mathcal{T}_i$ et $\mathcal{T}_j$ (il y a $O(|\mathcal{L}|)$ paquets, donc $O(|\mathcal{L}|^2)$ couples à considérer) et toute combinaison de couples de sommets v_i, v'_i de $\mathcal{T}_i$ et de couples de sommets v_j, v'_j de $\mathcal{T}_j$. Une implémentation naïve conduit donc à une complexité de $O(|\mathcal{L}|^6)$. Néanmoins, il n'est pas difficile de remarquer que, étant donnés deux paquets $\mathcal{T}_i$ et $\mathcal{T}_j$, on peut se contenter de considérer les couples $v_i, v'_i \in \mathcal{T}_i$ et $v_j, v'_j \in \mathcal{T}_j$ tels qu'il n'existe aucun w vérifiant (i) $w \in \mathcal{T}_i$ et $n(v_i) < n(w) < n(v'_i)$ ou (ii) $w \in \mathcal{T}_j$ et $n(v_j) < n(w) < n(v'_j)$, car, si $n(v_i) < n(v_j) < n(v'_i) < n(v'_j)$ et si un tel w existait, on aurait (iii) $w \in \mathcal{T}_i$ et soit $n(v_i) < n(v_j) < n(w) < n(v'_j)$, soit $n(w) < n(v_j) < n(v'_i) < n(v'_j)$, ou (iv) $w \in \mathcal{T}_j$ et soit $n(v_i) < n(w) < n(v'_i) < n(v'_j)$, soit $n(v_i) < n(v_j) < n(v'_i) < n(w)$: on pourrait donc remplacer v_i, v'_i, v_j ou v'_j par w.

Donc, en réalité, on a seulement $O(|\mathcal{L}|)$ couples de terminaux à considérer dans $\mathcal{T}_i$ (les couples de terminaux v_i, v'_i avec $n(v_i) < n(v'_i)$ tels qu'il n'existe aucun $w \in \mathcal{T}_i$ vérifiant $n(v_i) < n(w) < n(v'_i)$) et $O(|\mathcal{L}|)$ couples de terminaux à considérer dans $\mathcal{T}_j$. Enfin, étant donnés $v_i, v'_i \in \mathcal{T}_i$ et $v_j, v'_j \in \mathcal{T}_j$ (avec $n(v_i) < n(v'_i)$ et $n(v_j) < n(v'_j)$), décider si v_i, v'_i, v_j, v'_j vérifient la propriété (6.1) peut se faire en testant si $n(v_i) < n(v'_i) < n(v_j) < n(v'_j)$, ou si $n(v_j) < n(v'_j) < n(v_i) < n(v'_i)$, ou si $n(v_i) < n(v_j) < n(v'_j) < n(v'_i)$, ou si $n(v_j) <$

$n(v_i) < n(v'_i) < n(v'_j)$. En tout, on a donc $O(|\mathcal{L}|^2 \cdot |\mathcal{L}| \cdot |\mathcal{L}|) = O(|\mathcal{L}|^4)$ comparaisons à effectuer pour décider si la propriété (6.1) est vérifiée.

Enfin, on peut réduire la taille des instances de CMTM à résoudre grâce à la proposition suivante :

Proposition 6.1. *Soit une instance planaire de MCM et soit une solution optimale pour cette instance. Considérons le partitionnement associé à cette solution. Soit un paquet $\mathcal{T}_i$ de ce partitionnement tel qu'aucun autre des paquets ne soit plus court que lui. Alors, tout sommet de $p(u_i, v_i)$ appartient à la composante connexe contenant $\mathcal{T}_i$ après avoir enlevé les arêtes de la solution, où $n(u_i) = \min_{w \in \mathcal{T}_i} n(w)$ et $n(v_i) = \max_{w \in \mathcal{T}_i} n(w)$.*

Démonstration. Supposons qu'un sommet w de $p(u_i, v_i)$ ne soit pas dans la composante connexe contenant $\mathcal{T}_i$. Alors, la solution considérée étant optimale, w appartient à la composante connexe contenant $\mathcal{T}_j$ pour un $j \neq i$. Comme $\mathcal{T}_j$ n'est pas plus court que $\mathcal{T}_i$, il existe $v_j \in \mathcal{T}_j$ tel que $n(v_j) < n(u_i)$ ou $n(v_j) > n(v_i)$. À cause de la planarité du graphe, toute chaîne reliant u_i à v_i partage au moins un sommet avec toute chaîne reliant v_j à w : ceci contredit le fait que la solution considérée est admissible. □

Ainsi, pour un tel $\mathcal{T}_i$, on peut contracter les sommets de $p(u_i, v_i)$ en un seul sommet avant de résoudre l'instance de CMTM associée.

6.4 Bilan et problèmes ouverts

Dans ce chapitre, nous montrons que MCM est polynomial dans les graphes planaires, si tous les terminaux sont sur la face extérieure et s'ils sont en nombre fixé (alors qu'il est APX-difficile dans les étoiles non pondérées si le nombre de terminaux est quelconque). Nous décrivons également une approche que nous avons suivie pour tenter de résoudre MCM dans les graphes planaires ayant un nombre fixé de terminaux (mais n'ayant pas nécessairement tous leurs terminaux sur la face extérieure), et exposons comment cette approche nous a conduit à déceler une erreur dans l'algorithme conçu par Yeh et résolvant CMTM dans ce cas [164].

La première question qui reste en suspens est bien évidemment la suivante : quelle est la complexité de MCM dans les graphes planaires ayant un nombre fixé de terminaux (CMTM étant polynomial dans ce cas) ? En outre, comme nous l'avons montré dans ce chapitre, les différentes approches utilisées pour CMTM ne semblent pas pouvoir facilement s'adapter à MCM : donc, si ce problème est polynomial, il sera sûrement nécessaire d'explorer d'autres pistes pour le démontrer.

Une autre question à considérer est l'existence (ou non) d'un PTAS pour CMTM dans les graphes planaires : on sait que le problème est $\mathcal{NP}$-difficile [54], mais il n'a pas (encore ?) été montré APX-difficile. Cependant, bien

que CMTM soit polynomial dans les graphes planaires à k niveaux (puisque, d'après [54], il est polynomial dans les graphes de largeur d'arbre bornée), la méthode générale pour concevoir des PTAS développée dans [12] et basée sur une décomposition du graphe planaire en un ensemble de graphes planaires à k niveaux (on calcule une solution optimale pour les graphes planaires à k niveaux, et on combine ensuite toutes ces solutions en une unique solution pour le graphe initial ; voir [12] pour de plus amples détails) ne peut pas être utilisée pour CMTM, puisque, lorsque l'on recolle tous les morceaux (c'est-à-dire les coupes multiterminales partielles) ensemble, on n'obtient pas nécessairement une coupe multiterminale pour la totalité du graphe.

Il serait également intéressant d'étudier la complexité de CMTM dans les graphes planaires, lorsque les terminaux se trouvent sur un nombre borné de faces (le nombre de terminaux n'étant pas borné). Rappelons que, récemment, CMTM a été montré polynomial lorsque tous les terminaux se trouvent sur la même face [42].

Enfin, une question ouverte depuis longtemps (voir [54]) est : CMTM est-il FPT dans les graphes planaires (si le paramètre considéré est le nombre de terminaux) ?

Troisième partie

Graphes généraux

Chapitre 7

Comparaison expérimentale d'heuristiques pour maximiser le multiflot entier

7.1 Préliminaires

Comme nous l'avons déjà évoqué dans le chapitre 1, la résolution de problèmes de multiflots entiers est au cœur de diverses problématiques industrielles. En d'autres termes, la question de la résolution efficace de ces problèmes se pose en pratique. Il semble que, pour garantir des temps de calcul raisonnables, il convienne, pour l'instant, d'écarter les méthodes de résolution exactes, et de se tourner vers des méthodes rapides fournissant de "bonnes" solutions approchées.

Nous avons regroupé, dans le chapitre 1, ces méthodes sous le terme générique "heuristiques". Le problème qui se pose à nous est donc le suivant : concevoir des heuristiques rapides et efficaces pour MCDA et MFEM. Kleinberg et Tardos remarquaient déjà, en 1995, le manque d'approches heuristiques disponibles pour ces problèmes [102]. Selon eux, l'heuristique la plus communément employée pour résoudre des instances réelles de MCDA est l'algorithme SPF décrit en section 1.3.3, mais, malgré sa rapidité, ses performances sur ces instances restent décevantes. Ils ont donc proposé une méthode plus astucieuse pour résoudre MCDA dans une famille très générale de graphes. Malheureusement, cet algorithme est relativement compliqué (à énoncer et à implémenter), et il repose sur une hypothèse de parité des capacités que ne vérifient pas nécessairement toutes les instances à résoudre.

Récemment, d'autres schémas généraux de résolution pour MCDA ont été proposés par Chekuri, Khanna et Shepherd. Ils représentent une avancée significative dans ce domaine, mais présentent trois inconvénients. Le premier est que certains ne s'appliquent qu'à des graphes planaires [35, 37, 40] ; le second est qu'ils nécessitent de résoudre la relaxation continue du problème,

ce qui peut considérablement augmenter les temps de calcul sur des grosses instances. Enfin, leurs mises en œuvres ne paraît pas évidente a priori.

Une approche plus simple à envisager serait de résoudre la relaxation continue du problème et d'effectuer ensuite une phase d'arrondi aléatoire (voir [140] et le chapitre 1) ; cependant, cela nécessiterait également de résoudre la relaxation continue, et la phase d'arrondi peut mener à des solutions non admissibles (une autre procédure pour les rendre admissibles sans trop les dégrader serait alors nécessaire). Remarquons que, dans tous les cas, la qualité des solutions produites par ces algorithmes est fortement conditionnée par la valeur du ratio d'intégrité. Une autre approche possible est d'utiliser des méta-heuristiques ; cependant, de la définition de "voisinages" pertinents pour les problèmes étudiés dépend l'efficacité de ces méthodes. Des approches similaires ont déjà été utilisées pour des problèmes de multichemins à coût minimum [49, 165] : il conviendrait d'abord de s'assurer que les voisinages définis dans ces travaux peuvent s'adapter à MFEM. En outre, ces approches ne permettent pas d'exploiter les idées qui, de façon intuitive, semblent conduire à des solutions intéressantes (comme le fait qu'utiliser des chemins courts est, la plupart du temps, bénéfique ; rappelons que cette idée est à la base de l'algorithme SPF).

Dans ce chapitre, nous proposons une heuristique (combinatoire) dérivée de l'algorithme $ACGT$ décrit au chapitre 2. Nous formulons une première version de cette heuristique (avec deux variantes), que nous comparons à (une variante de) SPF (ces deux algorithmes ayant l'énorme avantage de pouvoir facilement être implémentés) sur des instances aléatoires de MFEM (sections 7.2 et 7.3). Nous formulons ensuite une deuxième version de l'heuristique en combinant $ACGT$ et SPF (section 7.4), et, finalement, nous concluons en analysant les résultats expérimentaux (section 7.5) et en faisant le bilan de leurs enseignements (section 7.6).

Dans la suite de ce chapitre, nous ne considérerons que des graphes connexes (s'ils ne le sont pas, on considère chaque composante connexe indépendamment) et non orientés.

7.2 Une première heuristique

La première heuristique que nous proposons pour MFEM est quasiment identique à $ACGT$:

- Construire un arbre couvrant de poids *maximum* ;
- Ensuite, utiliser l'algorithme de Garg et al. [80] pour router un multiflot entier sur cet arbre couvrant.

Remarquons que la dernière phase de l'algorithme $ACGT$ (pendant laquelle on construit une multicoupe pour la totalité du graphe) est ici inutile.

Nous appellerons $\mathcal{H}1$ cette heuristique : on peut faire deux remarques la concernant. D'abord, il existe des algorithmes très rapides et très simples à implémenter pour chacune de ses deux phases (nous détaillerons ce point ultérieurement). Ensuite, il est peu probable qu'elle donne des solutions de bonne qualité sur des instances quelconques (sauf peut-être dans des graphes ayant un nombre cyclomatique très petit), car elle laisse de côté beaucoup d'arêtes qui pourraient être utilisées pour router du flot. Cette deuxième remarque nous amène à énoncer une variante de $\mathcal{H}1$, que nous désignerons par $\mathcal{H}'1$:

- Itérer $\mathcal{H}1$ tant que la valeur du flot augmente dans le graphe (à chaque itération, on supprime du graphe les arêtes saturées et on diminue les capacités des arêtes du nombre d'unités de flot qui les traversent) ;
- Renvoyer la somme des flots calculés aux différentes itérations.

Ainsi, même s'il n'est pas difficile de montrer que $\mathcal{H}1$ et $\mathcal{H}'1$ ont le même comportement dans le pire cas (si l'on considère, par exemple, la famille d'instances donnée en figure 2.1) et que $\mathcal{H}'1$ est au moins aussi performante que $\mathcal{H}1$ (concernant la qualité de ses solutions), on peut espérer que, en pratique, $\mathcal{H}'1$ délivre des solutions de bien meilleure qualité que $\mathcal{H}1$, puisqu'elle continue d'utiliser des arêtes tant qu'il en reste suffisamment. La crainte que l'on peut légitimement avoir concerne le temps d'exécution de cette nouvelle heuristique : dans le pire cas, une seule arête peut être saturée à chaque itération, et on doit donc exécuter $O(m)$ fois l'heuristique $\mathcal{H}1$ (m étant le nombre d'arêtes). Les temps d'exécution de $\mathcal{H}1$ et $\mathcal{H}'1$ ne seront donc probablement pas comparables.

Nous donnons à présent le détail de notre protocole expérimental.

7.3 Réalisation des tests : algorithmes et implémentation

Notre objectif est à présent de comparer les performances de nos deux heuristiques $\mathcal{H}1$ et $\mathcal{H}'1$ à celles de SPF (voir la description de cet algorithme en section 1.3.3). Plus précisément, nous considérons une variante de SPF adaptée à MFEM : à chaque itération, on choisit le plus court chemin disponible (comme dans SPF), et, lorsqu'un chemin est choisi, on route le maximum d'unités de flot dessus (ce qui nécessite de trouver le minimum des capacités des arêtes composant ce chemin et de router un nombre d'unités de flot égal à cette quantité).

Les tests ont été réalisés en langage C [100]. L'intégralité du code source (environ 600 lignes de code) est disponible à l'adresse suivante :

http://cedric.cnam.fr/∼bentz/these/Sources_tests.html

Les instances utilisées pour ces tests ont été générées aléatoirement par le logiciel *Rudy* [144] ; lorsque cela était nécessaire, elles ont ensuite été rendues connexes à l'aide d'une procédure aléatoire très simple. En outre, les instances ont été générées en suivant les quelques règles suivantes :

- 4 valeurs de n (le nombre de sommets) ont été considérées : 50, 150, 175 et 200 ;
- 4 valeurs de m (le nombre d'arêtes) ont été considérées : $\frac{n(n-1)}{40}$, $\frac{3n(n-1)}{40}$, $\frac{n(n-1)}{8}$ et $\frac{n(n-1)}{4}$ (rappelons qu'un graphe complet compte $\frac{n(n-1)}{2}$ arêtes, et que ces valeurs correspondent donc à des graphes ayant respectivement 5%, 15%, 25% et 50% du nombre maximum d'arêtes) ;
- 2 valeurs de $|\mathcal{L}|$ ont été considérées : $\frac{n}{10}$ et $\frac{n}{5}$;
- Les capacités ont été tirées aléatoirement (par *Rudy*) dans l'intervalle $\{1, \ldots, c_{\max}\}$, où $c_{\max} = 10 + alea$, $alea$ étant un nombre généré aléatoirement par notre programme et compris entre 1 et $\frac{n}{10}$; l'objectif était d'obtenir des capacités assez petites, car le problème tend à devenir plus "simple" lorsque les capacités sont grandes [140] ;
- Pour chaque triplet $(n, m, |\mathcal{L}|)$ différent, on a généré 20 instances et fait la moyenne des résultats sur ces 20 instances (on a donc généré 640 instances en tout).

Il convient de noter que l'on a choisi des graphes ayant un nombre modéré d'arêtes, car, dans les graphes ayant beaucoup d'arêtes (les graphes complets, par exemple), la longueur moyenne des chemins a tendance à diminuer : dans ces graphes, l'algorithme SPF a très certainement un bien meilleur comportement que nos deux heuristiques. Cependant, on ne génère jamais d'arbres (car $m \geq \frac{n(n-1)}{40} > n - 1$ si $n > 40$).

Pour chaque instance, on a calculé une solution à l'aide de $\mathcal{H}1$, $\mathcal{H}'1$ et (la variante de) SPF. On a également calculé (pour information) la valeur optimale de la relaxation continue à l'aide du logiciel CPLEX 9.0 [95], et plus précisément en utilisant l'interface AMPL, qui permet de faire résoudre à CPLEX des PL formulés pratiquement sous leur forme naturelle (c'est-à-dire à l'aide de formules mathématiques). Nous ne décrirons pas ici le format utilisé par AMPL : nous renvoyons le lecteur intéressé à [68]. Néanmoins, l'un des avantages d'AMPL est qu'il sépare le modèle (fichier décrivant la forme générale du PL) des données (chaque fichier de données correspondant à une instance). En outre, une fois le fichier du modèle écrit (le modèle utilisé étant très simple dans notre cas ; nous le décrivons dans le paragraphe suivant), écrire les fichiers de données (c'est-à-dire les différentes instances) est une tâche relativement aisée en C, car le format diffère peu du format utilisé par *Rudy* pour générer les instances.

Pour calculer la valeur optimale de la relaxation continue du problème, on fournit à AMPL un modèle adapté aux graphes non orientés. En effet, nous avons vu en section 1.3.3 que le cas non orienté pouvait se réduire au cas

orienté, et se formuler ainsi à l'aide du modèle (IPL-MFEM-1), par exemple. Néanmoins, la transformation utilisée implique que le graphe orienté obtenu a cinq fois plus d'arcs que le graphe non orienté initial n'a d'arêtes (voir la figure 1.4). Le modèle que nous allons utiliser est plus compact : on remplace chaque arête (u, v) par deux arcs opposés (u, v) et (v, u) de même capacité ; la nouvelle contrainte de capacité porte alors sur la somme des flots traversant ces deux arcs (A désigne l'ensemble d'arcs obtenu). Voici ce modèle :

$$
\textbf{(IPL-MFEM-NO)} \quad \left|\begin{array}{lll}
\max & \displaystyle\sum_{i=1}^{|\mathcal{L}|} \sum_{(s_i,u)\in A} f^i_{s_i,u} & \\
\text{s. c.} & \displaystyle\sum_{(u,v)\in A} f^i_{u,v} = \sum_{(v,w)\in A} f^i_{v,w} & \forall i \in \{1,\ldots,|\mathcal{L}|\}, \forall v \in S \setminus \{s_i, s'_i\} \\
& \displaystyle\sum_{i=1}^{|\mathcal{L}|} \left(f^i_{u,v} + f^i_{v,u}\right) \leq c(u,v) & \forall (u,v) \in A \\
& \displaystyle\sum_{i=1}^{|\mathcal{L}|} \sum_{(u,s_i)\in A} f^i_{u,s_i} = 0 & \qquad (7.1) \\
& \displaystyle\sum_{i=1}^{|\mathcal{L}|} \sum_{(s'_i,u)\in A} f^i_{s'_i,u} = 0 & \qquad (7.2) \\
& f^i_{u,v} \in \mathbb{N} & \forall i \in \{1,\ldots,|\mathcal{L}|\}, \forall (u,v) \in A
\end{array}\right.
$$

La contrainte (7.1) assure qu'aucune unité du i^e flot quittant la i^e source n'y reviendra (la seule destination possible d'une telle unité est donc le puits correspondant, à cause des contraintes de conservation de flot en tous les autres sommets du graphe) ; de même, la contrainte (7.2) assure qu'aucune unité du i^e flot arrivant au i^e puits n'en repartira. Pour obtenir la relaxation continue que l'on souhaitait, il suffit à présent de relâcher les contraintes d'intégrité sur les $f^i_{u,v}$; on peut ensuite fournir les instances à résoudre à AMPL et CPLEX.

Nous décrivons à présent l'implémentation de SPF. Le déroulement détaillé de cet algorithme est le suivant :

Tant que non STOP :
- Pour chaque liaison (s_i, s'_i), on calcule le plus court chemin (s'il existe) entre s_i et s'_i en effectuant un *parcours en largeur d'abord* à partir de s_i [46] ;
- S'il existe, on garde le chemin le plus court parmi les (au plus $|\mathcal{L}|$) chemins calculés, sinon STOP ;
- On parcourt ce chemin pour trouver l'arête de capacité minimum et on route sur ce chemin un nombre d'unités de flot égal à cette capacité ; on met à jour les capacités et on supprime les arêtes saturées.

Remarquons qu'il est tout à fait possible que l'on puisse réutiliser des informations des itérations précédentes pour éviter de recalculer tous les chemins d'une itération à l'autre. Néanmoins, la topologie du graphe change d'une itération à l'autre (puisque les arêtes saturées sont retirées) ; cela ne paraît donc pas si simple à mettre en œuvre. Nous avons décidé de privilégier la simplicité, et de ne pas garder trace de ces informations ; le temps d'exécution ne semble pas trop en souffrir.

Intéressons-nous à présent à $\mathcal{H}1$. Le déroulement de cette heuristique a été implémenté ainsi :

- Calculer un arbre couvrant de poids maximum à l'aide d'une variante de l'algorithme de Kruskal (détaillée ci-dessous) ;
- Enraciner l'arbre couvrant en un sommet quelconque, c'est-à-dire, choisir un sommet r, puis calculer la hauteur des autres sommets dans l'arborescence de racine r en effectuant un parcours en largeur d'abord à partir de r [46] ;
- Pour chaque liaison (s_i, s'_i), calculer le chemin de s_i à s'_i dans l'arbre couvrant en effectuant un parcours en largeur d'abord à partir de s_i ; pour chaque i, $lca(i)$ (pour *least common ancestor* en anglais) est le sommet le plus "bas" dans le chemin de s_i à s'_i ;
- Pour chaque palier de l'arborescence, du plus haut au plus bas, et pour chaque liaison (s_i, s'_i) contenue dans une sous-arborescence enracinée en un sommet du palier courant (c'est-à-dire, $lca(i)$ appartient à ce palier), on route le maximum d'unités de flot entre s_i et s'_i, puis on met à jour les capacités résiduelles.

Enfin, l'implémentation de $\mathcal{H}'1$ à partir de celle de $\mathcal{H}1$ est immédiate.

Détaillons à présent la variante de l'algorithme de Kruskal [115] utilisée. Elle peut s'énoncer ainsi :

- Trier les arêtes dans l'ordre décroissant des poids et les placer (dans cet ordre) dans un tableau ; poser $i = 1$;
- Tant que non STOP : si ajouter la i^e arête du tableau au graphe en construction ne crée pas de cycle, le faire ; si on a ajouté $n - 1$ arêtes alors STOP, sinon poser $i := i + 1$.

La première étape est réalisée à l'aide d'un algorithme de *tri par segmentation* (*quicksort* en anglais). Ce n'est pas le meilleur algorithme de tri connu, mais il reste très efficace dans la majorité des cas, et est assez simple à implémenter. Il consiste, en substance, à choisir un *pivot* dans le tableau à trier, et à diviser les éléments du tableau en deux groupes : ceux qui sont inférieurs au pivot et que l'on place à sa gauche, et ceux qui lui sont supérieurs et que l'on place à sa droite. On recommence ensuite cette opération avec

les deux sous-tableaux situés respectivement à gauche et à droite du pivot ; on obtient ainsi une description récursive de cet algorithme. Pour plus de détails, se reporter à [46]. Bien qu'il existe des librairies implémentant cet algorithme en langage C, nous avons décidé de l'inclure dans notre code, pour le rendre auto-suffisant.

La deuxième étape de l'algorithme de Kruskal utilise une procédure simple dite de *fusion* : on associe un numéro à chaque composante connexe et, initialement, on a un numéro par sommet (et on a donc n composantes connexes). Lorsque l'on veut ajouter une arête, on vérifie que ses deux extrémités ont un numéro différent (et sont donc dans deux composantes connexes différentes) ; si c'est le cas, on ajoute l'arête, et on renumérote tous les sommets de la nouvelle composante connexe obtenue en utilisant le numéro d'une des deux extrémités de l'arête.

Nous sommes à présent en mesure d'effectuer nos tests. Ils ont été réalisés sur un serveur UNIX doté de 2 Go de mémoire vive, d'un processeur cadencé à 2,8 GHz et du logiciel CPLEX 9.0. Tous les résultats ne sont pas détaillés, pour la raison suivante. Il s'est avéré que, sur nos instances aléatoires, SPF fonctionnait très bien (et l'écart entre la valeur de la solution qu'il renvoie et la valeur optimale de la relaxation continue est souvent très faible). $\mathcal{H}1$, comme nous le pensions, s'est révélée très rapide (la plus rapide des trois) et de piètre qualité. Quant à $\mathcal{H}'1$, son temps d'exécution est beaucoup trop important par rapport à celui de SPF et (surtout) $\mathcal{H}1$, alors qu'elle renvoie des solutions de qualité comparable à celles de SPF. Les résultats concernant SPF et $\mathcal{H}1$ sont reportés en section 7.5. Devant ces constatations, nous avons cherché à élaborer une autre heuristique, plus efficace : $\mathcal{H}2$.

7.4 Une deuxième heuristique

Nous décrivons à présent l'heuristique $\mathcal{H}2$. Elle est fondée sur une idée simple : l'algorithme SPF est très efficace tant qu'il reste des chemins très courts. Par contre, lorsqu'il n'y a que des chemins "longs", router suivant des plus courts chemins peut s'avérer un très mauvais choix (voir, par exemple, la famille d'instances détaillée en section 2.1.2). Par ailleurs, l'heuristique $\mathcal{H}'1$ se comporte assez mal lorsqu'il y a beaucoup de chemins courts, car elle a besoin de beaucoup d'itérations pour saturer toutes les arêtes de ces chemins (puisqu'elle ne considère qu'un ensemble très restreint d'arêtes à chaque itération), mais elle se comporte bien mieux lorsque tous les chemins sont longs (en particulier parce que, si tous les chemins sont suffisamment longs, alors le graphe a moins d'arêtes et est donc plus proche d'un arbre).

Ainsi, $\mathcal{H}2$ prend en paramètre un entier λ et combine SPF et $\mathcal{H}'1$ de la façon suivante :

- Tant qu'il reste des chemins de longueur au plus λ, appliquer SPF ;
- Ensuite, appliquer $\mathcal{H}'1$ sur le graphe résiduel.

Bien sûr, il convient de déterminer de façon empirique une valeur pertinente pour le paramètre λ (ni trop grande, ni trop petite). En effet, si λ est trop grand, $\mathcal{H}2$ consiste simplement à appliquer SPF (si aucun chemin de longueur strictement supérieure à λ n'est utilisé par SPF) ; si λ est trop petit, la première phase de $\mathcal{H}2$ sera peu utile (elle laissera beaucoup d'arêtes à considérer dans la deuxième phase). Des premiers tests ont permis de conclure que la valeur $\lceil \log_{10}(n) \rceil$ était un bon compromis. Remarquons que cette quantité vaut 2 lorsque n est compris entre 11 et 100, et 3 lorsque n est compris entre 101 et 1000. Les résultats expérimentaux ont montré que $\mathcal{H}2$ améliorait sensiblement les performances de $\mathcal{H}'1$; en outre, elle améliore celles de SPF sur certains points (mais les dégrade sur d'autres). Nous détaillons l'ensemble de ces résultats dans la section suivante.

7.5 Résultats

Dans cette section, nous présentons et analysons les résultats des tests décrits dans les sections précédentes. Les quatre tableaux suivants correspondent aux résultats obtenus pour des graphes ayant 50, 150, 175 et 200 sommets respectivement (soit quatre valeurs de n différentes). Rappelons que chaque valeur indiquée dans une case de ces tableaux correspond à une moyenne sur vingt instances. En outre, les temps y sont donnés en secondes. Nous détaillons à présent les intitulés des lignes de chacun des tableaux :

- *% Nb. arêtes max.* est le nombre d'arêtes des instances générées, exprimé comme un pourcentage du nombre maximum d'arêtes $\frac{n(n-1)}{2}$;
- *Valeur PL* est la valeur optimale moyenne de la relaxation continue du problème pour les instances considérées ;
- *Temps PL* est le temps moyen mis par CPLEX pour calculer la valeur optimale de la relaxation continue du problème pour les instances considérées ;
- *Valeur SPF* est la valeur moyenne des solutions retournées par SPF pour les instances considérées (le pourcentage donné entre parenthèses est l'écart relatif entre *Valeur PL* et *Valeur SPF*) ;
- *Temps SPF* est le temps moyen mis par SPF pour calculer la solution renvoyée, pour les instances considérées ;
- *Long. max. SPF* est la longueur moyenne du plus long chemin examiné par SPF et ajouté dans la solution renvoyée, pour les instances considérées. Cette information a été ajoutée afin de comparer cette valeur à la valeur de λ utilisée dans $\mathcal{H}2$;
- *Valeur* $\mathcal{H}1$ (resp. *Valeur* $\mathcal{H}2$) est la valeur moyenne des solutions retour-

nées par $\mathcal{H}1$ (resp. $\mathcal{H}2$) pour les instances considérées (l'écart relatif entre *Valeur PL* et *Valeur $\mathcal{H}2$* est donné entre parenthèses) ;
- *Temps* $\mathcal{H}1$ (resp. *Temps* $\mathcal{H}2$) est le temps moyen mis par $\mathcal{H}1$ (resp. $\mathcal{H}2$) pour calculer la solution renvoyée, pour les instances considérées.

% Nb. arêtes max.	5		15		25		50	
$\|\mathcal{L}\|$	5	10	5	10	5	10	5	10
Valeur PL	47,4	58,8	192,2	303,8	338,4	576,6	724,6	1257,2
Temps PL	$<$ 0,1 s.	0,2 s.	$<$ 0,1 s.	0,8 s.	0,4 s.	2,2 s.	1 s.	4,2 s.
Valeur SPF	45,8 (3,38 %)	57 (3,06 %)	189,2 (1,56 %)	288,4 (5,07 %)	328 (3,07 %)	546,2 (5,27 %)	711,8 (1,77 %)	1210,2 (3,74 %)
Temps SPF	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.
Long. max. SPF	9,4	9,8	6,8	7,6	5,8	6,6	5	5,2
Valeur $\mathcal{H}1$	19,8	31,2	28,2	38,6	29,8	45,8	35	55
Temps $\mathcal{H}1$	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.
Valeur $\mathcal{H}2$	40,8 (13,92 %)	57 (3,06 %)	170,2 (11,45 %)	243,6 (19,82 %)	287,8 (14,95 %)	424,6 (26,36 %)	647 (10,71 %)	927,6 (26,22 %)
Temps $\mathcal{H}2$	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	0,2 s.

TAB. 7.1 – Résultats expérimentaux pour $n = 50$ ($\lambda = 2$).

% Nb. arêtes max.	5		15		25		50	
$\|\mathcal{L}\|$	15	30	15	30	15	30	15	30
Valeur PL	742,2	1276,8	1583,7	2199,1	2192,3	3834,6	5312,5	8137,8
Temps PL	$>$ 15 s.	$>$ 15 s.	$>$ 15 s.	$>$ 15 s.	$>$ 15 s.	$>$ 15 s.	$>$ 15 s.	$>$ 15 s.
Valeur SPF	718,6 (3,18 %)	1136,8 (10,96 %)	1464,6 (7,52 %)	2086,6 (5,12 %)	2065,4 (5,79 %)	3688,2 (3,82 %)	5150,6 (3,05 %)	7839 (3,67 %)
Temps SPF	0,2 s.	0,4 s.	0,6 s.	1,2 s.	1,4 s.	2,8 s.	2,8 s.	7,6 s.
Long. max. SPF	8	10	6	6,4	5	6	4,8	5,2
Valeur $\mathcal{H}1$	60,6	72	62	77,6	69,4	79,2	82,6	97
Temps $\mathcal{H}1$	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.	$<$ 0,1 s.
Valeur $\mathcal{H}2$	573,4 (22,74 %)	790,4 (38,1 %)	1248,8 (21,15 %)	1589 (27,74 %)	1950,6 (11,02 %)	2997 (21,84 %)	5046 (5,02 %)	7363,4 (9,52 %)
Temps $\mathcal{H}2$	$<$ 0,1 s.	0,4 s.	0,6 s.	1 s.	1,4 s.	2,6 s.	6,8 s.	13,2 s.
Val. SPF (Tps. $\mathcal{H}2$)	286,3	1136,8	1464,6	1572,7	2065,4	3005,6	5150,6	7839
Tps. SPF (Val. $\mathcal{H}2$)	$>$ 0,1 s.	0,3 s.	0,5 s.	1 s.	$>$ 1,3 s.	2,6 s.	2,8 s.	7 s.

TAB. 7.2 – Résultats expérimentaux pour $n = 150$ ($\lambda = 3$).

% Nb. arêtes max.	5		15		25		50	
$\|\mathcal{L}\|$	17	35	17	35	17	35	17	35
Valeur PL	760,4	1272,4	2498,3	4087	4538,9	6504,7	9176,8	13789,2
Temps PL	> 45 s.	> 45 s.	> 45 s.	> 45 s.	> 45 s.	> 45 s.	> 45 s.	> 45 s.
Valeur SPF	716,4 (5,79 %)	1121,4 (11,87 %)	2272,2 (9,05 %)	3874 (5,21 %)	4223,8 (6,94 %)	6246,6 (3,97 %)	8996,8 (1,96 %)	13490,8 (2,16 %)
Temps SPF	0,2 s.	1,2 s.	1,4 s.	3,8 s.	2,4 s.	6,4 s.	7 s.	15,2 s.
Long. max. SPF	8	10,6	5,4	6,2	5	5,4	5	5,4
Valeur $\mathcal{H}1$	68,8	79,6	74	97	77	102	78	105
Temps $\mathcal{H}1$	< 0,1 s.	< 0,1 s.	< 0,1 s.	< 0,1 s.	< 0,1 s.	< 0,1 s.	< 0,1 s.	< 0,1 s.
Valeur $\mathcal{H}2$	607,4 (20,12 %)	778,6 (38,8 %)	2015,8 (19,31 %)	2891 (29,26 %)	3788,2 (16,54 %)	5204,2 (19,99 %)	8851,4 (3,55 %)	12579,4 (8,77 %)
Temps $\mathcal{H}2$	0,2 s.	0,5 s.	0,4 s.	1,2 s.	2 s.	5 s.	14 s.	34 s.
Val. SPF (Tps. $\mathcal{H}2$)	760,4	314,1	310,6	1203	3504,6	5210,6	8996,8	13490,8
Tps. SPF (Val. $\mathcal{H}2$)	0,2 s.	1 s.	1,2 s.	2,1 s.	2,2 s.	5 s.	6,6 s.	13,6 s.

TAB. 7.3 – Résultats expérimentaux pour $n = 175$ ($\lambda = 3$).

% Nb. arêtes max.	5		15		25		50	
$\|\mathcal{L}\|$	20	40	20	40	20	40	20	40
Valeur PL	784	1147	3945,6	5998,3	6376,1	10124,6	12903,7	20132,9
Temps PL	> 120 s.	> 120 s.	> 120 s.	> 120 s.	> 120 s.	> 120 s.	> 120 s.	> 120 s.
Valeur SPF	733,4 (6,45 %)	1028,6 (12,28 %)	3751,4 (4,92 %)	5676,4 (5,37 %)	6091,6 (4,46 %)	9819,4 (3,01 %)	12558,4 (2,68 %)	19381,2 (3,73 %)
Temps SPF	0,8 s.	1,8 s.	2,4 s.	6,4 s.	9 s.	12 s.	11 s.	30,2 s.
Long. max. SPF	7,2	8,4	5,8	6,2	5,6	5,8	4,8	4,8
Valeur $\mathcal{H}1$	38,8	77,2	90,8	124,2	91,4	125,6	98	132,4
Temps $\mathcal{H}1$	< 0,1 s.	< 0,1 s.	< 0,1 s.	< 0,1 s.	< 0,1 s.	< 0,1 s.	< 0,1 s.	0,2 s.
Valeur $\mathcal{H}2$	570,4 (27,24 %)	768,8 (32,97 %)	3179,2 (19,42 %)	4191,8 (30,12 %)	5497,8 (13,77 %)	7776,2 (23,19 %)	12438,6 (3,6 %)	18243 (9,39 %)
Temps $\mathcal{H}2$	0,2 s.	0,4 s.	2 s.	2,6 s.	8,8 s.	9,8 s.	27,6 s.	53,8 s.
Val. SPF (Tps. $\mathcal{H}2$)	140,8	260,4	2936,4	2941,8	5994,2	8518,8	12558,4	19381,2
Tps. SPF (Val. $\mathcal{H}2$)	0,6 s.	1 s.	2,1 s.	3,4 s.	8 s.	8,2 s.	11 s.	27,4 s.

TAB. 7.4 – Résultats expérimentaux pour $n = 200$ ($\lambda = 3$).

Nous n'avons pas cherché à résoudre le problème en nombres entiers à l'aide de CPLEX, pour deux raisons. D'abord, cette résolution prend énormément de temps ; ensuite, la borne fournie par la relaxation continue s'est avérée suffisamment bonne pour nos tests.

Examinons à présent nos résultats. La première chose à remarquer est que, pour les graphes ayant 50 sommets, SPF est aussi (voire plus) rapide que $\mathcal{H}2$, et fournit des solutions sensiblement meilleures. Pour les graphes à 150, 175 et 200 sommets, nous avons ajouté les informations suivantes dans nos tableaux :

- *Val. SPF (Tps. $\mathcal{H}2$)* est la valeur moyenne des solutions retournées par SPF au bout du temps mis par $\mathcal{H}2$ pour retourner une solution, pour les instances considérées ;
- *Tps. SPF (Val. $\mathcal{H}2$)* est le temps moyen mis par SPF pour renvoyer une solution de même valeur que celle renvoyée par $\mathcal{H}2$, pour les instances considérées.

En effet, pour un nombre d'arêtes modéré (au plus 25 % de $\frac{n(n-1)}{2}$), on peut observer dans les trois tableaux de résultats 7.2, 7.3 et 7.4, que, en moyenne, SPF renvoie des solutions (sensiblement) meilleures que celles de $\mathcal{H}2$, mais est moins rapide que $\mathcal{H}2$. Plus précisément, en moyenne, l'heuristique $\mathcal{H}2$ est environ 30 % plus rapide que SPF, mais les solutions qu'elle renvoie sont 20 % moins bonnes. Il nous a donc semblé intéressant de répondre aux deux questions suivantes : si on se fixe une limite de temps (le temps mis par $\mathcal{H}2$ pour renvoyer une solution), les performances de SPF sont-elles meilleures que celles de $\mathcal{H}2$? Inversement, si on se fixe une valeur (celle de la solution renvoyée par $\mathcal{H}2$), SPF est-elle plus rapide que $\mathcal{H}2$ pour calculer une solution de cette valeur ?

D'après nos résultats, la réponse à chacune de ces questions est non pour $n = 175$ et $m \leq \frac{n(n-1)}{8}$ et pour $n = 200$ et $m \leq \frac{3n(n-1)}{40}$, et ce, quelle que soit la valeur (parmi celles considérées) du nombre de liaisons (dans les autres cas, le temps d'exécution de $\mathcal{H}2$ devient trop important). En d'autres termes, dès lors que l'on a $n \geq 175$ et $m \leq \frac{3n(n-1)}{40}$, $\mathcal{H}2$ est intéressante par rapport à SPF (et inversement) : l'une est plus rapide, l'autre fournit de meilleures solutions. Mais $\mathcal{H}2$ trouve plus vite que SPF une "bonne" solution (c'est-à-dire une solution dont la valeur est celle de la solution renvoyée par $\mathcal{H}2$), et on ne peut donc pas imaginer concurrencer $\mathcal{H}2$ à l'aide de SPF en arrêtant l'exécution de cette heuristique au bout d'un certain temps. Comme notre objectif était de concevoir une heuristique rapide pour MFEM et que l'écart relatif entre les valeurs des solutions renvoyées par les deux algorithmes est raisonnable (de l'ordre de 20 % en moyenne), les performances de $\mathcal{H}2$ nous paraissent acceptables (si on se restreint aux graphes ayant au moins 150 sommets et un nombre modéré d'arêtes).

Une autre remarque importante est que l'écart relatif entre la valeur de la solution renvoyée par SPF (et, à une échelle un peu moindre, celui entre la valeur de la solution renvoyée par $\mathcal{H}2$) et la valeur optimale de la relaxation continue est très faible ; en d'autres termes, le saut d'intégrité est faible sur les instances générées (alors, que, théoriquement, il peut être très grand). En outre, le temps nécessaire au calcul de cette borne supérieure est très important par rapport aux temps de calcul des deux heuristiques testées, ce qui confirme le fait que, pour les instances considérées, utiliser une méthode d'arrondi aléatoire pour obtenir une "bonne" solution entière n'est pas très intéressant (en terme de rapidité de calcul). De façon similaire, toutes les méthodes approchées présentées en section 7.1 qui s'appuient sur cette relaxation continue souffriront de ce handicap : il serait intéressant de déterminer si, dans ces conditions, utiliser une méthode rapide (de telles méthodes sont connues dans la littérature, voir [149] par exemple) fournissant une valeur approchée de cette borne permettrait d'améliorer sensiblement l'intérêt d'une telle approche (c'est-à-dire, en d'autres termes, permettrait de diminuer fortement les temps d'exécution sans trop dégrader la valeur de la borne).

Enfin, il convient de noter que, dans $\mathcal{H}2$, pour les cas $n = 175, m \leq \frac{n(n-1)}{8}$, et $n = 200, m \leq \frac{3n(n-1)}{40}$, nous limitons l'utilisation de l'algorithme SPF aux chemins de longueur $\lambda = \lceil \log_{10}(n) \rceil = 3$, alors que SPF est amené à examiner des chemins de longueur 6,6 (en moyenne). Dans ce cas, réduire la longueur des chemins examinés de 55 % permet donc d'obtenir un gain en temps de 30 % en moyenne.

7.6 Bilan et problèmes ouverts

La première conclusion à tirer des expérimentations présentées dans ce chapitre est que, sur les instances générées, les deux meilleures heuristiques sont SPF et $\mathcal{H}2$, chacune ayant ses propres avantages et inconvénients. En particulier, comme son principe le laissait présager, $\mathcal{H}2$ n'est compétitive que lorsque le nombre d'arêtes n'est pas trop élevé.

Néanmoins, un fait troublant est que, en moyenne, la valeur des solutions renvoyées par SPF est (sensiblement) meilleure que celle des solutions renvoyées par $\mathcal{H}2$. Bien que cette dernière soit plus rapide que SPF dans un certain nombre de cas (c'est son intérêt principal, et c'est la deuxième conclusion qui s'impose à nous), la stratégie qui consiste à router les flots sur un arbre couvrant de poids maximum semble donc moins bonne (en terme de qualité des solutions) que celle qui consiste à router les flots sur des plus courts chemins, même lorsqu'il ne reste plus de chemins "courts". En théorie, pourtant, des exemples où cette constatation n'est pas vérifiée sont connus (voir le début du chapitre 2).

Un deuxième fait troublant est que, sur les instances générées par *Rudy*, nous avons constaté que le saut d'intégrité était très faible (la valeur de la solution renvoyée par SPF étant très proche de l'optimum de la relaxation continue). Encore une fois, en théorie, on sait qu'il existe une infinité de graphes où ce n'est pas le cas. Le choix de se servir de *Rudy* pour effectuer nos tests provient du fait que c'est l'un des générateurs aléatoires de graphes les plus utilisés, mais cela ne garantit évidemment pas la pertinence des instances ainsi générées.

La première suite à donner à cette étude pourrait donc consister à tester $\mathcal{H}2$ sur des instances réelles (issues de problèmes industriels), pour lesquelles on sait, d'une part, que SPF donne des résultats médiocres (d'après [102]), et, d'autre part, que le saut d'intégrité est relativement élevé. Il serait également intéressant de poursuivre des tests sur des instances beaucoup plus grosses, car on peut s'apercevoir, à la lumière des résultats de notre étude, que, en terme de temps de calcul, l'heuristique $\mathcal{H}2$ ne semble devenir vraiment intéressante par rapport à SPF que lorsque la taille des instances générées atteint un certain seuil.

La seconde consisterait évidemment à concevoir de nouvelles heuristiques pour MFEM, soit en imaginant et en exploitant de nouvelles idées, soit en découvrant un moyen original de combiner les idées sur lesquelles se fondent les quelques algorithmes approchés déjà connus pour ce problème [41, 102, 103].

Chapitre 8

Flot multiterminal dans les graphes orientés

8.1 Préliminaires

Dans ce chapitre, nous étudions le problème FMTEM dans les graphes orientés. En particulier, nous donnons les premiers résultats de complexité le concernant, et nous introduisons un paramètre k_S pour ce problème, pour lequel nous fournissons une caractérisation complète des cas "faciles" et "difficiles". Nous l'utilisons également pour améliorer l'algorithme approché de Garg et al. Nous commençons par rappeler quelques résultats préliminaires.

Tout d'abord, CMTM, le problème de coupe associé à FMTEM, a été largement étudié dans le cas non orienté, aussi bien du point de vue de l'approximation que de la résolution exacte en temps polynomial [26, 42, 52, 54, 91, 97, 164]. Le cas orienté de CMTM a aussi fait l'objet de quelques papiers : Garg et al. [78] montrent qu'il est APX-difficile même pour $|\mathcal{T}| = 2$ (rappelons que $\mathcal{T}$ est l'ensemble des terminaux) et donnent un algorithme $2\log_2|\mathcal{T}|$-approché (que nous détaillons ci-dessous, car nous aurons à le réutiliser), et Naor et Zosin [130] donnent un algorithme 2-approché. Cependant, l'algorithme de Garg et al. a une propriété intéressante : il calcule une coupe multiterminale dont la valeur est au plus $2\log_2|\mathcal{T}|$ fois la valeur d'un flot multiterminal entier, et est donc un algorithme $2\log_2|\mathcal{T}|$-approché à la fois pour CMTM et pour FMTEM (alors que l'algorithme de Naor et Zosin ne fournit pas une solution approchée pour FMTEM). En outre, Costa et al. [52] montrent que FMTEM et CMTM sont polynomiaux dans les GSC, en utilisant une simple réduction (que nous détaillons ci-dessous, car elle sera un des composants essentiels de nos algorithmes) à des problèmes de flot maximum et de coupe minimum, respectivement. À notre connaissance, ce sont les seuls résultats concernant FMTEM dans les graphes orientés. Dans les graphes non orientés, FMTEM a récemment été montré polynomial via la méthode des ellipsoïdes [99] : la résolution est fondée sur le théorème de

Mader sur les $\mathcal{T}$-chemins [151, Chapitre 73].

***ABS*, un algorithme $2\log_2|\mathcal{T}|$-approché pour FMTEM.** L'algorithme conçu par Garg et al. pour FMTEM suit une approche primale-duale remarquablement simple. On suppose sans perte de généralité que le nombre de terminaux est un multiple de 2 (si ce n'est pas le cas, on ajoute des terminaux fictifs). Le déroulement de l'algorithme est le suivant :

- Pour i de 1 à $\log_2|\mathcal{T}|$, partitionner les terminaux de chaque composante connexe du graphe mis à jour (pour $i = 1$, c'est le graphe initial) en deux groupes de même taille (appelés respectivement *premier* et *deuxième* groupe), puis calculer une coupe minimum et un flot maximum entre l'union des premiers groupes et l'union des deuxièmes groupes (en reliant une source virtuelle à tous les terminaux des premiers groupes et tous les terminaux des deuxièmes groupes à un puits virtuel), puis faire de même entre l'union des deuxièmes groupes et l'union des premiers groupes. Conserver le flot maximum parmi ces deux flots et mettre à jour le graphe ;
- Lorsque $i = \log_2|\mathcal{T}|$, renvoyer le flot de valeur maximum parmi tous ceux calculés lors des $\log_2|\mathcal{T}|$ itérations.

Nous appellerons cet algorithme *ABS*, pour *Algorithme par Bissections Successives*. Remarquons que le flot renvoyé est effectivement un flot multiterminal valide et entier, puisqu'il a été obtenu à l'aide d'un algorithme de flot maximum (on suppose évidemment que les capacités sont entières). Il nous reste à montrer le théorème suivant :

Théorème 8.1 ([78]). *ABS est $2\log_2|\mathcal{T}|$-approché pour FMTEM.*

Démonstration. Il n'est pas difficile de voir qu'on obtient une coupe multiterminale valide en faisant l'union des $2\log_2|\mathcal{T}|$ coupes minimums calculées aux $\log_2|\mathcal{T}|$ itérations (on calcule deux coupes minimums à chaque itération). Le flot multiterminal entier renvoyé par l'algorithme étant le meilleur des $2\log_2|\mathcal{T}|$ flots maximaux associés à ces coupes, et la valeur d'un flot maximum dans un graphe étant égale à la valeur d'une coupe minimum, on obtient une solution admissible pour FMTEM dont la valeur est au plus $2\log_2|\mathcal{T}|$ fois la valeur d'une solution pour CMTM, ce qui termine la preuve. □

Polynomialité de FMTEM et CMTM dans les GSC. Nous détaillons à présent la transformation utilisée par Costa et al. dans [52] pour réduire FMTEM et CMTM dans les GSC à des problèmes de flot maximum et de coupe minimum respectivement. L'idée est d'appliquer l'opération suivante (que nous appellerons opération *EST*, pour opération par *Éclatement et Substitution de Terminaux*) pour $\bar{\mathcal{T}} = \mathcal{T}$ (c'est-à-dire, à tous les terminaux).

Opération EST (donnée : un ensemble de terminaux $\bar{\mathcal{T}}$) :
- Pour chaque terminal $t_i \in \bar{\mathcal{T}}$, remplacer t_i par deux nouveaux sommets (non terminaux) t'_i et t''_i, et remplacer tout arc (v, t_i) par un arc (v, t'_i) (ayant la même capacité) et tout arc (t_i, v) par un arc (t''_i, v) (ayant la même capacité) ;
- Ajouter deux nouveaux sommets terminaux σ et τ, et relier chaque t'_i à τ et σ à chaque t''_i (les arcs ajoutés doivent avoir des capacités "suffisamment grandes").

La figure 8.1 illustre cette opération, qui servira de base à nos algorithmes.

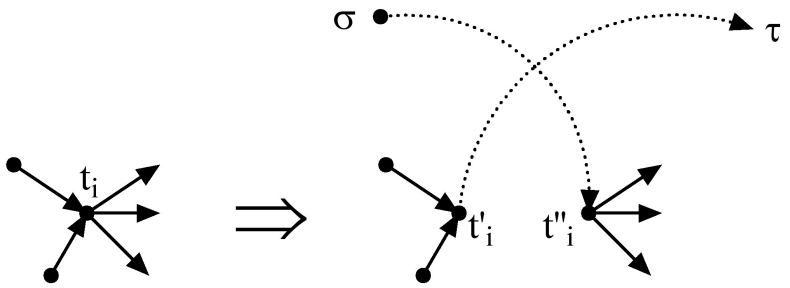


FIG. 8.1 – Transformation du terminal t_i.

On calcule ensuite une coupe minimum et un flot maximum entre σ et τ. Il n'est pas difficile de voir que la coupe obtenue est une coupe multiterminale valide pour l'instance initiale. En outre, le flot obtenu est un flot multiterminal valide si aucune unité de flot n'est routée entre t''_i et t'_i pour tout i. Dans ce cas, la valeur du flot multiterminal obtenu étant égale à la valeur de la coupe multiterminale, les deux sont optimaux.

Costa et al. ont montré le théorème suivant :

Théorème 8.2 ([52]). *FMTEM et CMTM sont polynomiaux dans les GSC.*

Démonstration. Il suffit d'appliquer l'opération EST pour $\bar{\mathcal{T}} = \mathcal{T}$, puis de calculer un flot maximum et une coupe minimum entre σ et τ. Il n'existe aucun chemin entre t''_i et t'_i pour tout i, le graphe étant sans circuits : le flot obtenu est donc un flot multiterminal valide, ce qui conclut la preuve. □

Dans toute la suite de ce chapitre, nous généralisons les deux théorèmes précédents en introduisant un nouveau paramètre, et nous donnons une caractérisation des cas "faciles" et "difficiles" de FMTEM vis-à-vis de ce paramètre. Ces résultats ont fait l'objet d'une publication en revue internationale [17].

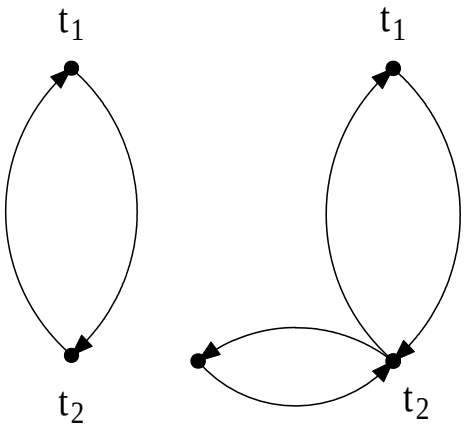
8.2 Définition du paramètre k_S

Étant donné un ensemble de terminaux $\mathcal{T}$, on définit un terminal **solitaire** comme étant un terminal qui se trouve sur au moins un circuit ne contenant pas d'autres terminaux. On note $\mathcal{T}_S \subseteq \mathcal{T}$ l'ensemble des terminaux solitaires, et on définit $k_S = |\mathcal{T}_S|$ (on a donc $k_S \leq |\mathcal{T}|$).

Remarquons que $k_S = 0$ dans les GSC, et que, plus généralement, la valeur de k_S est bornée par $|\mathcal{T}|$ d'une part et par le nombre de circuits d'autre part (ces deux quantités pouvant même être arbitrairement grandes devant k_S).

De façon intuitive, la condition $k_S = 0$ garantit que toute unité de flot routée entre t_i et l'un des $|\mathcal{T}|$ terminaux (en incluant t_i lui-même) passera à travers au moins un terminal différent de t_i : par conséquent, on peut considérer qu'elle est routée d'un terminal vers un autre terminal. Nous verrons un peu plus loin dans ce chapitre comment calculer k_S.

La figure 8.2 montre des exemples avec différentes valeurs de k_S.



(a) $k_S = 0$. (b) $k_S = 1$.

FIG. 8.2 – Différentes valeurs de k_S.

8.3 Preuve de APX-difficulté

Nous montrons dans cette section que FMTEM est APX-difficile dans les graphes orientés, même si $|\mathcal{T}| = k_S = 2$ (ou si $k_S = 1$ et $|\mathcal{T}| = 3$).

Pour cela, nous allons utiliser le fait que, si $\mathcal{P} \neq \mathcal{NP}$, il n'y a aucun algorithme ρ-approché pour MFEM dans les graphes orientés, même avec deux liaisons, pour tout $\rho < 2$ (rappelons que ce problème est APX-difficile même dans les graphes non orientés [89, p. 489]). Commençons par en donner une courte preuve.

Tout d'abord, rappelons que le problème CDA est $\mathcal{NP}$-complet dans les graphes orientés, même avec seulement deux liaisons [67] (une instance de CDA avec deux liaisons est dite *positive* si les deux chemins existent). Ce résultat a été utilisé dans [89] pour prouver un résultat d'inapproximabilité très fort concernant MFEM dans les graphes orientés. Soit $\mathcal{I}$ une instance orientée quelconque de CDA avec deux liaisons (s_1, s'_1) et (s_2, s'_2). On ramène $\mathcal{I}$ à une instance de MFEM en ajoutant deux sommets s''_1 et s''_2, deux arcs (s'_1, s''_1) et (s'_2, s''_2) de capacité 1, et en remplaçant les deux liaisons par les liaisons (s_1, s''_1) et (s_2, s''_2). S'il existe un algorithme approché pour MFEM avec un ratio $\rho < 2$, un tel algorithme fournirait une solution de valeur 1 si $\mathcal{I}$ n'est pas positive, et de valeur 2 sinon (puisque $\frac{2}{\rho} > 1$) : ceci donnerait un algorithme polynomial pour résoudre CDA. D'où :

Proposition 8.1. *Dans les graphes orientés, MFEM est $\mathcal{NP}$-difficile à approcher à un ratio $2 - \epsilon$ pour tout $\epsilon > 0$, même si $|\mathcal{L}| = 2$.*

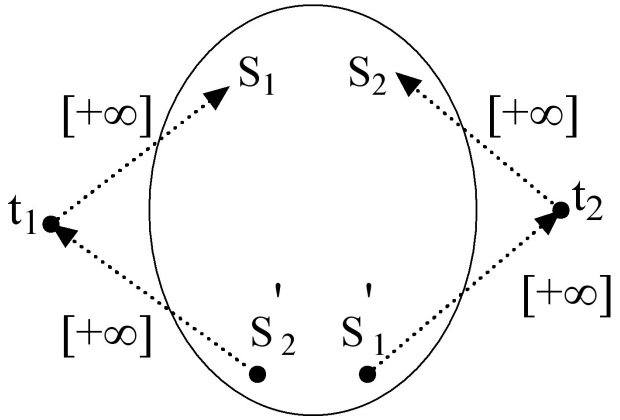
Il nous reste à montrer que, quand $|\mathcal{T}| = 2$ (c.-à-d., $\mathcal{T} = \{t_1, t_2\}$), FMTEM est *équivalent*, dans les graphes orientés, à MFEM avec deux liaisons (s_1, s'_1) et (s_2, s'_2). En effet, étant données (s_1, s'_1) et (s_2, s'_2), nous pouvons obtenir une instance équivalente de FMTEM en définissant deux nouveaux terminaux, t_1 et t_2, et en reliant (par des arcs ayant des capacités suffisamment grandes) t_1 à s_1, s'_2 à t_1, t_2 à s_2 et s'_1 à t_2 : toute unité de flot routée entre t_1 et t_2 est routée de s_1 à s'_1 ou de s_2 à s'_2. Réciproquement, étant donnés deux terminaux t_1 et t_2, on peut définir $s_1 = t_1$, $s'_1 = t_2$, $s_2 = t_2$ et $s'_2 = t_1$. Nous venons donc de montrer (voir aussi la figure 8.3(a)) :

Théorème 8.3. *Dans les graphes orientés, FMTEM avec deux terminaux est équivalent à MFEM avec deux liaisons.*

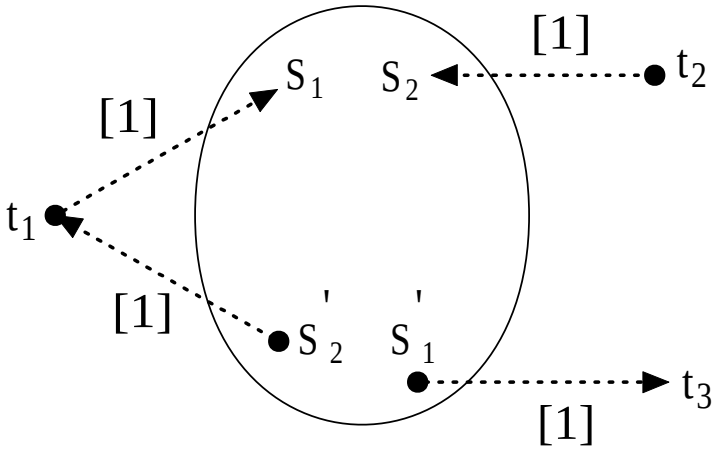
Évidemment, cette transformation ne peut pas être utilisée dans le cas non orienté, ni avec plus de deux terminaux. Cela implique :

Théorème 8.4. *Dans les graphes orientés, FMTEM est $\mathcal{NP}$-difficile à approcher à un ratio $2 - \epsilon$ pour tout $\epsilon > 0$, même lorsque $k_S = |\mathcal{T}| = 2$.*

Notons que cette borne inférieure de 2 est fine pour $|\mathcal{T}| = 2$, puisque l'algorithme ABS est 2-approché dans ce cas. En outre, ce résultat est pour l'essentiel comparable au résultat d'inapproximabilité (c.-à-d., APX-difficulté) connu pour le problème de coupe associé CMTM dans les graphes orientés [78]. Maintenant, nous prouvons le même résultat pour le cas $|\mathcal{T}| = 3$, $k_S = 1$. La preuve est très similaire : étant donnée une instance de CDA avec deux liaisons (s_1, s'_1) et (s_2, s'_2), on définit trois nouveaux terminaux t_1, t_2, t_3, et on ajoute quatre arcs de poids 1, $(t_1, s_1), (s'_2, t_1), (t_2, s_2), (s'_1, t_3)$ (voir la figure 8.3(b)) ; alors, tout chemin quittant t_1 est routé vers t_3. Donc, s'il existe deux chemins, il s'agit nécessairement d'un chemin entre t_1 et t_3 et d'un chemin entre t_2 et t_1. Notons que $\mathcal{T}_S = \{t_1\}$, puisque le seul terminal se trouvant sur un circuit est t_1. Par conséquent :



(a) Réduire MFEM à FMTEM pour $k_S = 2$ et $|\mathcal{T}| = 2$.



(b) Réduire CDA à FMTEM pour $k_S = 1$ et $|\mathcal{T}| = 3$.

FIG. 8.3 – Réductions pour l'APX-difficulté de FMTEM.

Théorème 8.5. *Dans les graphes orientés, FMTEM est $\mathcal{NP}$-difficile à approcher à un ratio $2-\epsilon$ pour tout $\epsilon > 0$, même lorsque $|\mathcal{T}| = 3$ et $k_S = 1$.*

Nous traitons le cas $k_S = 0$ (qui généralise le cas sans circuits) en section 8.4 et le cas $k_S = 1$ et $|\mathcal{T}| = 2$ en section 8.5.

8.4 Algorithmes exacts et approchés pour FMTEM

D'après la section précédente, FMTEM est APX-difficile pour $k_S \geq 1$ dans les graphes orientés. Par conséquent, si $\mathcal{P} \neq \mathcal{NP}$, les seuls algorithmes efficaces que l'on peut espérer concevoir pour le cas $k_S \geq 1$ sont des algorithmes approchés. Dans cette section, nous améliorons l'algorithme $2\log_2|\mathcal{T}|$-approché de Garg et al. et donnons un algorithme $2\log_2(k_S+2)$-approché pour FMTEM dans les graphes orientés. Nous montrons aussi que le cas $k_S = 0$ (qui généralise le cas sans circuits) est polynomial.

Nous commençons par détailler le second résultat. La première chose à remarquer est que l'on peut obtenir un renforcement intéressant de [52, proposition 3] (ce résultat est résumé dans le théorème 8.2) en notant que, s'il n'y a aucun terminal solitaire, alors, en appliquant l'opération EST à *tous* les terminaux (c'est-à-dire, pour $\bar{\mathcal{T}} = \mathcal{T}$), il ne subsiste aucun chemin orienté de t_i'' à t_i' pour chaque i, et par conséquent on peut résoudre FMTEM et CMTM par la même approche que celle utilisée par Costa et al. dans [52].

Théorème 8.6. *CMTM et FMTEM sont polynomiaux dans les graphes orientés si $k_S = 0$, en utilisant un simple algorithme de coupe minimum et de flot maximum.*

La figure 8.4 montre une application de cette méthode à l'instance de la figure 8.2(a).

En fait, si on veut garantir que, après application de l'opération EST, le graphe modifié n'admet pas de chemin de t_i'' à t_i' pour un i (autrement, on ne peut pas être sûr que le flot que l'on calculera dans le graphe modifié sera un flot multiterminal valide pour le graphe initial), c'est pour l'essentiel la meilleure (c'est-à-dire, la plus faible) hypothèse que l'on puisse faire.

Théorème 8.7. *Après avoir appliqué l'opération EST à tous les terminaux, il n'y a aucun chemin orienté de t_i'' à t_i' pour chaque i si et seulement si $k_S = 0$.*

Démonstration. Immédiate d'après les définitions de $\mathcal{T}_S$ et de l'opération EST. □

Les idées décrites précédemment nous fournissent une méthode pour calculer k_S (et $\mathcal{T}_S$) : on applique l'opération EST à tous les terminaux, et on vérifie, pour chaque terminal $t_i \in \mathcal{T}$, l'appartenance de t_i à $\mathcal{T}_S$ en testant s'il existe un chemin de t_i'' à t_i' dans le graphe modifié. Les théorèmes 8.6 et

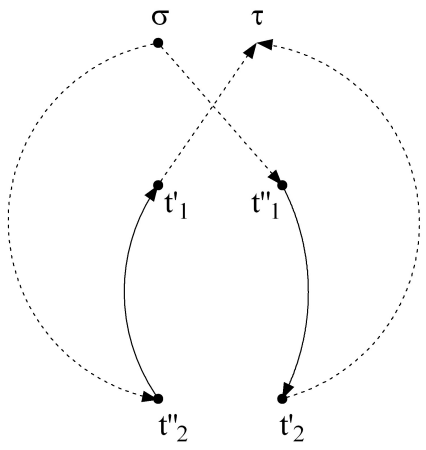


FIG. 8.4 – Transformer une instance de FMTEM en un problème de flot maximum.

8.7 montrent aussi l'importance du paramètre k_S pour FMTEM et CMTM. En particulier, ils suggèrent d'utiliser l'approche suivante pour trouver des solutions approchées pour ces deux problèmes, qui peut être vue comme une combinaison originale des résultats présentés dans [52] et dans [78].

Algorithme $EST - ABS$:

1. Si $|\mathcal{T}| \geq k_S + 2$, appliquer l'opération EST avec $\bar{\mathcal{T}} = \mathcal{T} \setminus \mathcal{T}_S$;
2. Calculer une solution pour la nouvelle instance obtenue (c'est-à-dire, où l'ensemble des terminaux est $\mathcal{T}_S \bigcup \{\sigma, \tau\}$) à l'aide de ABS.

Par conséquent, la différence principale avec l'algorithme ABS est que, avant d'utiliser une stratégie de "diviser-pour-régner", on "remplace" les terminaux dans $\mathcal{T} \setminus \mathcal{T}_S$ par seulement deux terminaux, σ et τ. Cela implique que l'on utilise ABS sur une instance avec $k_S + 2$ terminaux, et on obtient donc un facteur d'approximation de $2\log_2(k_S + 2)$ (au lieu de $2\log_2|\mathcal{T}|$). Le point clé est que toute unité de flot routée lors de l'exécution de l'algorithme de σ à τ par l'intermédiaire de t_i'' et t_i' pour un certain i avec $t_i \notin \mathcal{T}_S$ (s'il en existe une) peut être reacheminée entre σ et un terminal dans $\mathcal{T}_S$ ou entre un terminal dans $\mathcal{T}_S$ et τ (car $t_i \in \mathcal{T} \setminus \mathcal{T}_S$). Ceci implique :

Théorème 8.8. *$EST - ABS$ est un algorithme $2\log_2(\min(k_S + 2, |\mathcal{T}|))$-approché pour FMTEM dans les graphes orientés.*

Notons d'ailleurs que l'opération EST peut être d'un intérêt indépendant, puisqu'elle peut toujours être appliquée afin de réduire la taille de l'instance (c.-à-d., pour ramener le nombre de terminaux de $|\mathcal{T}|$ à $k_S + 2$)

dans toute procédure calculant un flot multiterminal entier (par exemple, une procédure exacte par énumération implicite).

En fait, on peut montrer que notre analyse du ratio d'approximation de $EST - ABS$ est fine. Pour ce faire, nous utilisons un exemple construit à partir d'un arbre non orienté ayant 2^p sommets. Plus précisément, on se donne une chaîne $Q = u_0, u_1, \ldots, u_p$, et, pour chaque $i \in \{0, \ldots, p-1\}$, $2^i - 1$ feuilles sont reliées par une arête de capacité 1 à u_i (soit $\mathcal{F}_i$ cet ensemble de feuilles). Tous les sommets sont des sommets terminaux (et par conséquent $|\mathcal{T}| = 2^p$, c.-à-d., $p = \log_2 |\mathcal{T}|$), et les p arêtes de Q sont pondérées par un nombre entier N très grand (voir figure 8.5).

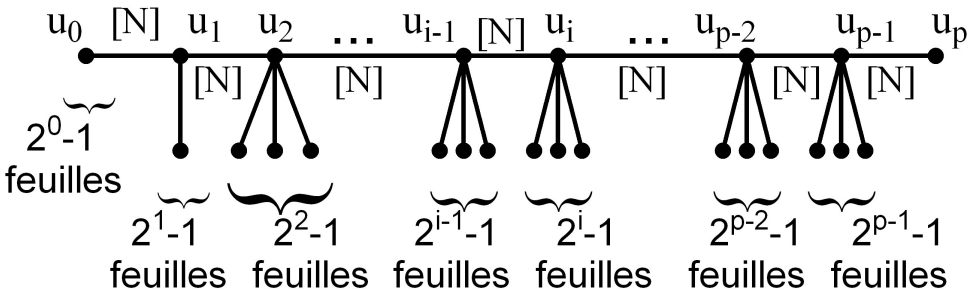


FIG. 8.5 – Une instance fine pour $EST - ABS$.

Pour transformer ce graphe non orienté en un graphe orienté, on remplace chaque arête par le "gadget" donné en figure 1.4 en section 1.3.3 : chaque arc du "gadget" a la même capacité que l'arête initiale. Dans cette instance, tous les terminaux sont solitaires (c.-à-d., $\mathcal{T} = \mathcal{T}_S$), et donc $EST - ABS$ consiste simplement à appliquer ABS. Par conséquent, nous allons prouver la finesse de l'analyse de Garg et al., et cela impliquera la finesse de la nôtre. On suppose sans perte de généralité que ABS fait toujours les mauvais choix lorsque plusieurs sont équivalents, et qu'il calcule une solution en séparant de façon itérative u_{p-i+1} et $\mathcal{F}_{p-i}$ de u_{p-i} : en effet, ceci va résulter en une recherche dichotomique sur $|\mathcal{T}|$. En outre, le premier flot maximum calculé à la i^e itération consistera à router N unités de flot entre u_0 et u_{p-i+1} et une unité de flot entre u_{p-i} et chaque feuille de $\mathcal{F}_{p-i}$ (le deuxième flot est symétrique ; il a donc la même valeur). La coupe minimum consistera à couper deux arcs différents (adjacents au même terminal) dans le "gadget" reliant u_{p-i} à u_{p-i+1} et dans le "gadget" reliant u_{p-i} à chaque feuille de $\mathcal{F}_{p-i}$. L'algorithme ABS fournit un flot de valeur au plus $N + (2^{p-1} - 1) + (2^{p-2} - 1) + \cdots + (2^1 - 1) = N + 2^p - p - 1$ (même s'il combine les flots obtenus lors des différentes itérations de la meilleure façon possible) et une coupe de valeur $2(pN + 2^p - p - 1)$. Le rapport entre ces deux valeurs est égal à $\frac{2(pN+2^p-p-1)}{N+2^p-p-1}$, et tend vers $2p = 2\log_2 k_S$ quand N augmente, ce qui établit la finesse de l'analyse de Garg et al. Cependant, cela ne signifie pas que l'on

ne peut pas espérer trouver un meilleur ratio d'approximation en utilisant un algorithme (même légèrement) différent.

Enfin, notons qu'un ratio plus fin peut être obtenu pour des cas particuliers (ainsi, si $k_S = 1$, notre analyse fournit un algorithme 2-approché).

8.5 Un cas facile : $k_S = 1$ et $|\mathcal{T}| = 2$

Dans les graphes orientés, FMTEM est APX-difficile même lorsque $|\mathcal{T}| = k_S = 2$ et lorsque $k_S = 1$ et $|\mathcal{T}| = 3$ (voir la section 8.3), et polynomial si $k_S = 0$ (voir la section 8.4). Dans cette section, nous traitons le dernier cas et prouvons que FMTEM est polynomial quand $k_S = 1$ et $|\mathcal{T}| = 2$.

Soit $\mathcal{T} = \{t_1, t_2\}$ et $\mathcal{T}_S = \{t_1\}$. D'après la définition de $\mathcal{T}_S$, il existe au moins un circuit contenant t_1 mais pas t_2, et il n'existe aucun circuit contenant t_2 mais pas t_1. Nous allons prouver que, en fait, cette instance peut être transformée en une instance équivalente où $k_S = 0$; puis, nous conclurons en utilisant le théorème 8.6. Nous montrons le lemme suivant :

Lemme 8.1. *Soit $\mathcal{I}$ une instance de FMTEM dans un graphe orienté avec $\mathcal{T} = \{t_1, t_2\}$ et $\mathcal{T}_S = \{t_1\}$. Sur tout circuit contenant t_1 mais pas t_2, il existe un arc* effaçable*, c'est-à-dire, un arc non utilisé par au moins une solution optimale pour $\mathcal{I}$.*

Démonstration. Soit $\mathcal{C} = \{t_1, u_1, u_2, \ldots, u_p, t_1\}$ un circuit ne contenant pas t_2. Nous prouvons que, sur $\mathcal{C}$, il existe un arc ne se trouvant ni sur un chemin élémentaire de t_1 à t_2 ni sur un chemin élémentaire de t_2 à t_1. Soit i tel qu'il y a un chemin de u_i à t_2 ne contenant pas t_1, mais il n'y a aucun chemin de u_{i+1} à t_2 ne contenant pas t_1. Si i existe, alors l'arc (u_i, u_{i+1}) peut être enlevé. En effet, il ne peut pas se trouver sur un chemin élémentaire de t_1 à t_2 (puisqu'il n'y a aucun chemin de u_{i+1} à t_2 ne contenant pas t_1), et il ne peut pas se trouver sur un chemin élémentaire de t_2 à t_1 (puisqu'autrement il existe un chemin de t_2 à u_i qui ne contient pas t_1, et $t_2 \in \mathcal{T}_S$). Si i n'existe pas, alors nous pouvons enlever (t_1, u_1) (s'il n'y a aucun chemin de u_1 à t_2 ne contenant pas t_1) ou (u_p, t_1) (s'il existe un chemin de u_p à t_2 qui ne contient pas t_1). Le lemme 8.1 est donc démontré. □

La preuve du lemme 8.1 fournit également un algorithme pour résoudre FMTEM dans ce cas-là, en enlevant des arcs de façon itérative jusqu'à ce qu'il ne reste aucun terminal solitaire. Il convient d'ailleurs de remarquer que tous les arguments utilisés dans cette preuve peuvent être testés en temps polynomial. En effet, étant donnés trois sommets u, v, w, tester s'il existe un chemin de u à v ne contenant pas w peut se faire en appliquant l'opération EST à w (ou bien en supprimant le sommet w) et en vérifiant dans le graphe obtenu s'il existe un chemin de u à v (à l'aide d'un simple parcours de graphe). De la même façon, on peut tester de façon efficace s'il existe un circuit passant par u mais pas par v, par exemple. En outre, notons

que l'approche utilisée pour résoudre FMTEM dans le cas $k_S = 1$ et $|\mathcal{T}| = 2$ est également valide pour CMTM.

Enfin, il n'est pas difficile de voir que les résultats de cette section permettent de déduire un cas polynomial pour MFEM (et pour MCM) dans les graphes orientés : lorsque $\mathcal{L} = \{(s_1, s'_1), (s_2, s'_2)\}$, s'il n'existe pas de chemin entre s_1 et s'_2 (ou, de façon symétrique, entre s_2 et s'_1), alors l'instance de FMTEM équivalente (obtenue à l'aide de la transformation de la figure 8.3(a)) vérifie $k_S \leq 1$, et peut donc être résolue par notre approche en temps polynomial.

8.6 Bilan et problèmes ouverts

Le paramètre k_S présenté dans ce chapitre permet d'améliorer notre connaissance de la frontière entre les cas "faciles" et "difficiles" de FMTEM dans les graphes orientés. Plusieurs résultats, positifs et négatifs, au sujet de la complexité et de l'approximabilité de ce problème, sont fournis. Les résultats de complexité sont dérivés de résultats connus pour MFEM et CDA, mais, à notre connaissance, n'existaient pas auparavant. Cependant, deux questions importantes restent ouvertes : existe-t-il un algorithme $O(1)$-approché pour le cas orienté général ? Existe-t-il d'autres cas particuliers "faciles" (par exemple, les graphes orientés planaires) ?

Chapitre 9

Conclusion générale

Les travaux exposés dans la présente thèse s'attachent principalement à l'étude de deux problèmes d'optimisation combinatoire liés, en particulier, au domaine des télécommunications. Nous nous sommes surtout intéressé aux aspects algorithmiques qui sont associés à ces problèmes. En effet, nous avons systématiquement tenté de répondre à une seule question : tel problème admet-il un algorithme de résolution efficace ? Cependant, bien que cette interrogation constitue un cadre général qui englobe tous les résultats que nous avons obtenus, un point important concerne les différentes significations de l'expression "un algorithme de résolution efficace". Comme nous l'avons déjà mentionné, tous les algorithmes que nous considérons sont polynomiaux, mais certains sont "exacts" (au sens où ils fournissent une solution optimale), d'autres sont "approchés" ; d'autres, encore, ne sont polynomiaux que lorsqu'un certain paramètre est fixé.

Il convient tout d'abord de noter que les problèmes principaux étudiés dans cette thèse, les problèmes du multiflot entier maximum et de la multicoupe minimum, sont connus pour être $\mathcal{NP}$-difficiles, et donc, selon toute vraisemblance, impossibles à résoudre de façon exacte à l'aide d'algorithmes efficaces, même dans des cas très particuliers. La recherche de cas polynomiaux demande donc d'être vigilant : il devient nécessaire de rendre les problèmes "moins difficiles" en considérant des hypothèses supplémentaires, qui peuvent nous aider à établir des propriétés structurelles sur lesquelles fonder des algorithmes ; d'un autre côté, il faut s'efforcer de ne pas ajouter trop d'hypothèses, pour ne pas aboutir à un cas trivial ou totalement inintéressant. De façon similaire, les cas où ces problèmes sont $\mathcal{NP}$-difficiles ne sont réellement intéressants que s'ils diffèrent peu de cas polynomiaux connus. En d'autres termes, il faut essayer de cerner au plus proche la frontière entre les cas "faciles" et les cas "difficiles". C'est ce que nous avons essayé de faire à chaque fois que nous nous sommes intéressé à cet aspect des problèmes.

Dans le chapitre 2, nous montrons que MCDA est polynomial dans les graphes de nombre cyclomatique borné. Ceci généralise et unifie les cas des anneaux et des arbres, les deux seules classes de graphes dans lesquelles ce problème était auparavant connu comme polynomial. En outre, nous avons vu que ce problème était APX-difficile dans les cactus, et qu'il était donc peu probable qu'il existe une classe plus générale que la nôtre où le problème reste polynomial.

Dans le chapitre 3, nous donnons un algorithme polynomial pour résoudre MCM dans les graphes de largeur d'arbre bornée, lorsque le nombre de liaisons est fixé. Il convient de remarquer, d'une part, que c'est une classe de graphes dans laquelle le sous-problème CMTM est polynomial, et, d'autre part, que MCM devient APX-difficile si le graphe est quelconque ou si le nombre de liaisons n'est pas fixé. Nous montrons également que MCM est $\mathcal{NP}$-difficile dans les GSC, alors que CMTM y est polynomial et que MCM est polynomial dans les arbres orientés.

Dans le chapitre 4, nous prouvons la polynomialité de MFEM dans les anneaux : ce problème est polynomial dans les chaînes et les étoiles, mais $\mathcal{NP}$-difficile dans les arbres.

Dans le chapitre 5, nous résolvons MCLM dans un canal dense pondéré ; ceci généralise les résultats de Formann et al. sur ce même problème. Nous résolvons également un cas particulier de MFEM et MCM dans des grilles uniformes, qui correspond à un cas particulier de CDA étudié par Frank. En outre, nous donnons des preuves de complexité montrant que des cas un peu plus généraux de MCM deviennent $\mathcal{NP}$-difficiles.

Dans le chapitre 6, nous montrons que MCM est polynomial dans les graphes planaires lorsque les terminaux sont sur la face extérieure et en nombre fixé (le cas des arbres étant déjà connu), alors qu'il est APX-difficile dans les étoiles non pondérées si $|\mathcal{L}|$ est quelconque et dans les graphes généraux si $|\mathcal{L}| = 3$.

Dans le dernier chapitre, nous donnons une caractérisation des cas "faciles" et "difficiles" de FMTEM dans les graphes orientés vis-à-vis d'un nouveau paramètre k_S que nous introduisons et étudions ; en particulier, nous montrons que le cas $k_S = 0$ est polynomial et généralise le cas des GSC étudié par Costa et al.

Par ailleurs, les travaux précédents concernaient, en majorité, l'approximation de ces problèmes. En examinant un peu l'ensemble de ces résultats, on comprend aisément pourquoi ces problèmes paraissent si compliqués à résoudre de façon algorithmique : dans de nombreux cas, concevoir un algorithme approché ayant un ratio raisonnable est déjà un problème $\mathcal{NP}$-difficile. Néanmoins, la classification en cas "faciles à approcher" et "difficiles à approcher" reste problématique et incomplète : pour beaucoup d'entre eux, on ignore ce qu'il en est réellement. Certains de nos résultats tentent donc de pallier partiellement ce manque. En particulier, on peut remarquer qu'il

n'existe que très peu de méthodes approchées qui ont été conçues pour les problèmes de multiflot entier ; ce sont souvent les mêmes algorithmes qui sont réutilisés pour obtenir différentes séries de résultats. Ainsi, lorsque l'on montre qu'il existe des familles d'instances parmi celles que l'on étudie où ces algorithmes donnent toujours de mauvais résultats, il reste peu de possibilités à exploiter. Nous avons donc également essayé de mettre au point de nouvelles méthodes algorithmiques.

Dans le chapitre 2, nous donnons des algorithmes approchés à ratios constants pour MFEM dans les graphes de nombre cyclomatique borné et pour MCDA dans les graphes planaires à k niveaux d'arêtes. On peut noter, d'une part, que ces deux classes de graphes généralisent les arbres (une des topologies les plus classiques), et, d'autre part, qu'il n'existe que très peu de classes de graphes où ces problèmes admettent des algorithmes approchés à ratios constants (on peut principalement citer les arbres et les grilles).

Dans le chapitre 3, nous montrons que le cas orienté de MCM se comporte différemment du cas non orienté dans les graphes non pondérés de largeur d'arbre et de degré maximum bornés, puisqu'il est APX-difficile alors que ce dernier admet un PTAS.

Dans le dernier chapitre, nous concevons un algorithme $2\log_2(k_S + 2)$-approché pour FMTEM dans les graphes orientés, améliorant ainsi l'algorithme de Garg et al. (au moins dans la majorité des cas, c'est-à-dire quand $|\mathcal{T}| > k_S + 2$). Nous montrons également que ce problème ne peut pas admettre d'algorithme approché ayant un ratio plus petit que 2.

Enfin, bien que les techniques utilisées dans nos preuves soient, en règle générale, plutôt classiques, elles sont relativement variées : réduction à un problème formulable par un PL dont la matrice est TU, méthodes primales-duales, combinaison d'énumération limitée et de programmation dynamique, raisonnement par récurrence, réduction à un problème de décision connexe, réduction à des problèmes polynomiaux basiques (comme le calcul d'un flot maximum), etc. Nous nous sommes également efforcé de réutiliser ou d'adapter autant que possible les résultats précédents ou les méthodes existantes, de façon à tirer parti des différentes recherches déjà menées sur ces sujets. En outre, le chapitre 2 présente, à nos yeux, deux intérêts particuliers : d'abord, à notre connaissance, l'algorithme approché de Garg et al. pour MFEM et MCM dans les arbres n'avait jamais été utilisé à l'intérieur d'un algorithme approché pour d'autres classes de graphes (alors que l'idée, qui paraît assez naturelle, a été utilisée avec succès pour d'autres types de problèmes) ; ensuite, l'introduction des graphes planaires à k niveaux d'arêtes nous semble pertinente, puisqu'ils généralisent les cactus, communément étudiés, et constituent un parallèle intéressant avec la classique famille des graphes planaires à k niveaux. De surcroît, ces deux points sont liés, puisque nous n'avons été en mesure d'utiliser efficacement l'algorithme de Garg et al. qu'en nous restreignant à des graphes planaires à k niveaux d'arêtes (si

l'on excepte les graphes de nombre cyclomatique borné).

Les tableaux C.1 et C.2 donnés en annexe C récapitulent les résultats les plus significatifs obtenus au cours de cette thèse, et les recadrent par rapport aux principaux résultats existants.

Il subsiste, cependant, de nombreux problèmes importants qui restent ouverts. Nous en avons sélectionné dix (parmi ceux énoncés dans les différents chapitres) et les avons listés en annexe C.2 ; nous ne nous étendrons pas davantage, car ils sont décrits plus en détails dans les chapitres concernés. Néanmoins, nous voudrions à présent décrire un autre problème ouvert qui n'est pas non plus, à nos yeux, dénué d'intérêt. Il concerne le ratio d'intégrité de MFEM dans les arbres : d'une part, Garg et al. ont montré qu'il était au plus 2 (tout comme celui de MCM) ; d'autre part, on ne connaît pas d'instance où ce ratio est plus grand que $\frac{3}{2}$. Le problème est donc le suivant : qu'en est-il en réalité ? Rappelons que, d'après la section 1.3.3, un exemple d'instance où ce ratio est égal à $\frac{3}{2}$ est donné par une étoile non pondérée à trois feuilles, où il existe une liaison entre chaque paire de feuilles. Le multiflot entier maximum vaut 1 dans ce graphe, alors que le multiflot continu maximum vaut $\frac{3}{2}$. En revanche, on sait (toujours d'après la section 1.3.3) que le ratio d'intégrité de MCM dans les arbres peut atteindre 2.

Pour conclure, on peut faire trois remarques à l'issue de cette étude.

La première est que nous n'avons toujours pas à notre disposition un panel conséquent de méthodes algorithmiques différentes pour résoudre de façon approchée les problèmes MFEM et MCDA. Depuis le début de cette thèse, et indépendamment de nos travaux, quelques progrès ont été effectués en ce sens, principalement par Chekuri, Khanna et Shepherd, qui ont développé un schéma général d'approximation qu'ils ont adapté à plusieurs cas. Leurs travaux sont remarquables car leurs idées, bien qu'issues en partie de travaux de Robertson et Seymour, sont originales dans le domaine de l'approximation des problèmes de multiflot entier ; en outre, leur application à ces problèmes et leur mise en œuvre est en soit un travail techniquement difficile. Ils ont toutefois détaillé les limites de leur approche : pour progresser encore, des idées encore plus novatrices et, selon toute vraisemblance, encore plus sophistiquées, seront nécessaires. Les limites des approches actuelles tiennent en particulier dans le fait qu'elles s'appuient sur des raisonnements où l'on compare la valeur d'une solution obtenue à l'optimum continu du problème (et non à l'optimum entier directement, car on ne sait pas le faire) : comme nous l'avons vu, l'écart entre les optima continu et entier peut être très grand même dans les graphes planaires, ce qui rend ces approches impuissantes à traiter les cas les plus difficiles. Un important travail de fond est donc indispensable, et nous pensons que cela prendra encore de nombreuses années pour obtenir des avancées significatives.

La seconde remarque est que nous ne nous sommes pas du tout inté-

ressé, dans cette thèse, aux méthodes générales de résolution exacte pour MFEM et MCM (comme les méthodes arborescentes décrites au chapitre 1, par exemple) : dans le chapitre 7, nous nous intéressons à la résolution pratique de MFEM à l'aide de méthodes heuristiques (et donc approchées) uniquement. Il suffit cependant de faire quelques tests à l'aide d'un logiciel comme CPLEX pour s'apercevoir rapidement que la résolution exacte de ces problèmes (et notamment des problèmes MFEM et MCDA) reste très problématique en pratique. Malgré tout, quasiment aucune étude n'a tenté de remédier à ce problème. Par exemple, à notre connaissance, la seule tentative de résolution de MFEM par ce type de méthodes est l'article de Brunetta et al., paru en 2000. Il y a, de toute évidence, un manque flagrant qu'il serait particulièrement intéressant de combler.

Enfin, la dernière remarque est que de nombreuses variantes de MFEM et MCM ont récemment été définies et étudiées. Certaines correspondent à des modèles de routage différents, ou bien reflètent plus fidèlement les avancées technologiques récentes dans le domaine des réseaux de télécommunications, et ont donc un intérêt pratique. D'autres, en revanche, constituent simplement des problèmes plus généraux, qui se posent dans d'autres contextes. Dans les deux cas, il serait profitable de déterminer si les résultats connus pour MFEM et MCM pourraient s'y appliquer, et comment modifier les travaux existants pour les adapter à ces nouveaux problèmes. En particulier, l'approximation des versions en ligne (*on-line* en anglais) de ces problèmes n'a été que peu abordée.

Bibliographie

[1] U. Adamy, C. Ambuehl, R. Sai Anand et T. Erlebach. Call Control in Rings. Actes ICALP, Lecture Notes in Computer Science 2380 (2002) 788–799.

[2] A. Agarwal, M. Charikar, K. Makarychev et Y. Makarychev. $O(\sqrt{\log n})$-approximation algorithms for Min UnCut, Min 2CNF Deletion, and directed cut problems. Actes STOC (2005).

[3] A. Aggarwal, A. Bar-Noy, D. Coppersmith, R. Ramaswani, B. Schieber et M. Sudan. Efficient routing and scheduling algorithms for optical networks. Actes SODA (1994) 412–423.

[4] A. Aggarwal, J. Kleinberg et D.P. Williamson. Node-disjoint paths on the mesh and a new trade-off in VLSI layout. Actes STOC (1996).

[5] R. Ahuja, T. Magnanti et J. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, New Jersey (1993).

[6] R. Sai Anand et T. Erlebach. Routing and Call Control Algorithms for Ring Networks. Actes WADS, Lecture Notes in Computer Science 2748 (2003) 186–197. (Également rapport technique TIK 171 (2003), 42 p. ETH Zurich.)

[7] M. Andrews, J. Chuzhoy, S. Khanna et L. Zhang. Hardness of the undirected edge-disjoint paths problem with congestion. Actes FOCS (2005).

[8] M. Andrews et L. Zhang. Hardness of the undirected edge-disjoint paths problem. Actes STOC (2005).

[9] S. Arora, C. Lund, R. Motwani, M. Sudan et M. Szegedy. Proof verification and the hardness of approximation problems. Journal of the ACM 45 (1998) 501–555.

[10] Y. Aumann et Y. Rabani. Improved bounds for all optical routing. Actes SODA (1995) 567–576.

[11] Y. Aumann et Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. SIAM Journal on Computing 27 (1998) 291–301.

[12] B.S. Baker. Approximation Algorithms for NP-complete Problems on Planar Graphs. J. ACM 41 (1994) 153–180.

[13] A. Baveja et A. Srinivasan. Approximation algorithms for disjoint paths and related routing and packing problems. Mathematics of Operations Research 25 (2000) 255–280.

[14] R. Bellman. Dynamic Programming. Princeton University Press (1957).

[15] C. Bentz. Edge disjoint paths and max integral multiflow/min multicut theorems in planar graphs. Actes ICGT (Hyères), Electronic Notes in Discrete Mathematics 22 (2005) 55-60.

[16] C. Bentz. Edge disjoint paths and multicut problems in graphs generalizing the trees. Rapport technique CEDRIC 948 (2005).
Page web : http://cedric.cnam.fr/AfficheArticle.php?id=948&lang=fr

[17] C. Bentz. The maximum integer multiterminal flow problem in directed graphs. À paraître dans Operations Research Letters (2006).

[18] C. Bentz, M.-C. Costa, C. Picouleau et M. Zrikem. The shortest multipaths problem in a capacitated dense channel. À paraître dans European Journal of Operational Research (2006).

[19] C. Bentz, M.-C. Costa et F. Roupin. Maximum integer multiflow and minimum multicut problems in two-sided uniform grid graphs. À paraître dans Journal of Discrete Algorithms (2006).

[20] C. Bentz, M.-C. Costa, L. Létocart et F. Roupin. Multicut and integral multiflow in rings. Rapport CEDRIC 1050 (2006), soumis. Version préliminaire : M.-C. Costa, L. Létocart et F. Roupin. Minimal multicut and maximal integer multiflow in rings. Actes ISMP (2003).
Page web : http://cedric.cnam.fr/PUBLIS/RC1050.pdf

[21] C. Berge. *Graphes et Hypergraphes*. Dunod, Paris (1969).

[22] D. Bertsimas, C.-P. Teo et R. Vohra. Analysis of LP relaxations for multiway and multicut problems. Networks 34 (1999) 102-114.

[23] H.L. Bodlaender. Planar graphs with bounded treewidth. Rapport technique RUU-CS-88-14, Utrecht University, The Netherlands (1988).

[24] U. Brandes, G. Neyer et D. Wagner. Edge-disjoint paths in planar graphs with minimum total length. Rapport technique, Konstanzer Schriften in Mathematik und Informatik 19, Universität Konstanz (1996), 11 p.

[25] L. Brunetta, M. Conforti et M. Fischetti. A polyhedral approach to an integer multicommodity flow problem. Discrete Applied Mathematics 101 (2000) 13–26.

[26] G. Călinescu, H. Karloff et Y. Rabani. An improved approximation algorithm for Multiway Cut. Journal of Computer and System Sciences 60 (2000) 564–574.

[27] G. Călinescu, C.G. Fernandes et B. Reed. Multicuts in unweighted graphs and digraphs with bounded degree and bounded tree-width. Journal of Algorithms 48 (2003) 333–359.

[28] P. Carmi, T. Erlebach et Y. Okamoto. Greedy edge-disjoint paths in complete graphs. Actes WG, LNCS 2880 (2003) 143–155. (Également rapport technique TIK 155 (2003).)

[29] A. Chakrabarti, C. Chekuri, A. Gupta et A. Kumar. Approximation Algorithms for the Unsplittable Flow Problem. Actes APPROX (2002) 51–66.

[30] W.-T. Chan et F.Y.L. Chin. Efficient algorithms for finding the maximum number of disjoint paths in grids. Journal of Algorithms 34 (2000) 337–369.

[31] S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani et D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. Actes CCC (2005).

[32] C. Chekuri et S. Khanna. Edge disjoint paths revisited. Actes SODA (2003) 628–637.

[33] C. Chekuri, M. Mydlarz et F.B. Shepherd. Demand Flow in a Tree and Packing Integer Programs. Actes ICALP (2003).

[34] C. Chekuri, S. Khanna et F.B. Shepherd. The All-Or-Nothing Multicommodity Flow Problem. Actes STOC (2004).

[35] C. Chekuri, S. Khanna et F.B. Shepherd. Edge-disjoint paths in planar graphs. Actes FOCS (2004).

[36] C. Chekuri, A. Gupta et A. Kumar. On a Bidirected Relaxation for the Multiway Cut Problem. Discrete Applied Mathematics 150 (2005) 67–79.

[37] C. Chekuri, S. Khanna et F.B. Shepherd. Multicommodity Flow, Well-linked Terminals, and Routing Problems. Actes STOC (2005).

[38] C. Chekuri, A. Gupta, I. Newman, Y. Rabinovich et A. Sinclair. Embedding k-Outerplanar Graphs into l_1. SIAM Journal on Discrete Math. 20 (2006) 119–136.

[39] C. Chekuri, S. Khanna et F.B. Shepherd. An $O(\sqrt{n})$-approximation and integrality gap for Disjoint Paths and UFP. Theory of Computing 2 (2006) 137–146.

[40] C. Chekuri, S. Khanna et F.B. Shepherd. Edge-Disjoint Paths in Planar Graphs with Constant Congestion. Actes STOC (2006).

[41] C. Chekuri, S. Khanna et F.B. Shepherd. A Note on Multicommodity Flows and Treewidth. Manuscrit, soumis (2006).
Page web : http://theory.cs.uiuc.edu/∼chekuri/papers/tw.ps

[42] D.Z. Chen et X. Wu. Efficient algorithms for k-terminal cuts on planar graphs. Algorithmica 38 (2004) 299–316. Version préliminaire : actes ISAAC (2001) 332–344.

[43] J. Cheriyan, H. Karloff et Y. Rabani. Approximating directed multicuts. Actes FOCS (2001) 320–328.

[44] S. Chopra et M.R. Rao. On the multiway cut polyhedron. Networks 21 (1991) 51–89.

[45] S.A. Cook. The complexity of theorem-proving procedures. Actes STOC (1971) 151–158.

[46] T. Cormen, C. Leiserson, R. Rivest et C. Stein. *Introduction to Algorithms (2nd edition)*. MIT Press (2001).

[47] S. Cosares et I. Saniee. An optimization problem related to balancing loads on SONET rings. Telecommunication Systems 3 (1994) 165–181.

[48] M.-C. Costa, A. Hertz et M. Mittaz. Bounds and Heuristics for the Shortest Capacitated Paths Problem. Journal of Heuristics 8 (2002) 449–466.

[49] M.-C. Costa, F.-R. Monclar et M. Zrikem. Variable Neighborhood Search for the Optimization of Cable Layout Problem. Journal of Intelligent Manufacturing 13 (2002) 353–365.

[50] M.-C. Costa, L. Létocart et F. Roupin. A greedy algorithm for multicut and integral multiflow in rooted trees. Operations Research Letters 31 (2003) 21–27.

[51] M.-C. Costa et A. Billionnet. Multiway cut and integer flow problems in trees. Actes CTW, ENDM 17 (2004) 105–109. Également actes CO'02 (2002).

[52] M.-C. Costa, L. Létocart et F. Roupin. Minimal multicut and maximal integer multiflow: A survey. EJOR 162 (2005) 55–69. Version préliminaire : actes ECCO XIV (2001).

[53] W.H. Cunningham. The optimal multiterminal cut problem. DIMACS Series in Disc. Math. and Theor. Comput. Sci. 5 (1991) 105–120.

[54] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour et M. Yannakakis. The complexity of multiterminal cuts. SIAM J. Comput. 23 (1994) 864–894.

[55] N. Derhy. Résolution pratique de problèmes de multicoupes dans les graphes non orientés. Mémoire d'ingénieur, CEDRIC-CNAM (2005).

[56] R.G. Downey et M.R. Fellows. *Parameterized Complexity.* Springer-Verlag, New York (1999).

[57] R.G. Downey et C. McCartin. Bounded Persistence Pathwidth. Actes CATS, CRPIT 41 (2005) 51–56.

[58] P.L. Erdös et L.A. Székely. Algorithms and Min-max Theorems for Certain Multiway Cuts. Actes IPCO (1992) 334–345.

[59] P.L. Erdös et L.A. Székely. On weighted multiway cuts in trees. Math. Programming 65 (1994) 93–105.

[60] P.L. Erdös, A. Frank et L.A. Székely. Minimum multiway cuts in trees. Discrete Applied Mathematics 87 (1998) 67–76.

[61] T. Erlebach. Approximation algorithms and complexity results for path problems in trees of rings. Actes MFCS, LNCS 2136 (2001) 351–362. (Également rapport technique TIK 109 (2001).)

[62] T. Erlebach et K. Jansen. The maximum edge-disjoint paths problem in bidirected trees. SIAM J. on Discrete Mathematics 14 (2001) 326–355.

[63] T. Erlebach et D. Vukadinovic. Path problems in generalized stars, complete graphs, and brick wall graphs. Discrete Applied Mathematics 154 (2006) 673–683. Version préliminaire : actes FCT (2001) 483–494.

[64] S. Even, A. Itai et A. Shamir. On the complexity of timetable and multicommodity flow problems. SIAM J. Comp. 5 (1976) 691–703.

[65] L.R. Ford et D.R. Fulkerson. Maximal Flow Through a Network. Canadian Journal of Mathematics 8 (1956) 339–404.

[66] M. Formann, D. Wagner et F. Wagner. Routing through a dense channel with minimum total wire length. Journal of Algorithms 15 (1993) 267–283.

[67] S. Fortune, J. Hopcroft et J. Wyllie. The directed subgraph homeomorphism problem. Theoretical Computer Science 10 (1980) 111–121.

[68] R. Fourer, D.M. Gay et B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming (2nd edition).* Duxbury Press, Brooks/Cole Publishing Company (2002). Page web : http://www.ampl.com

[69] A. Frank. Disjoint paths in a rectilinear grid. Combinatorica 2 (1982) 361–371.

[70] A. Frank. Edge-disjoint paths in planar graphs. J. Comb. Theory, Series B 39 (1985) 164–178.

[71] A. Frank et Z. Szigeti. A note on packing paths in planar graphs. Math. Programming 70 (1995) 201–210.

[72] A. Frank, A. Karzanov et A. Sebö. On integer multiflow maximization. SIAM J. Discrete Mathematics 10 (1997) 158–170.

[73] A. Frank, T. Ibaraki et H. Nagamochi. Two Arc-Disjoint Paths in Eulerian Digraphs. SIAM Journal on Discrete Mathematics 11 (1998) 557–589.

[74] A. Frank, B. Shepherd, V. Tandon et Z. Végh. Node-capacitated ring routing. Rapport technique EGRES 2001-09 (2001).

[75] A. Frieze. Edge-disjoint paths in expander graphs. SIAM Journal on Computing 30 (2000) 1790–1801.

[76] M.R. Garey et D.S. Johnson. Computers and Intractability: a Guide to the Theory of NP-completeness. W.H. Freeman and Co., San Fransisco (1979).

[77] N. Garg. Multicommodity Flows and Approximation Algorithms. PhD thesis, IIT, Delhi, Inde (1994).

[78] N. Garg, V.V. Vazirani et M. Yannakakis. Multiway cuts in directed and node weighted graphs. Actes ICALP, Lecture Notes in Computer Science 820 (1994) 487–498.

[79] N. Garg, V.V. Vazirani et M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. SIAM J. Comput. 25 (1996) 235–251.

[80] N. Garg, V.V. Vazirani et M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. Algorithmica 18 (1997) 3–20.

[81] N. Garg, V.V. Vazirani et M. Yannakakis. Multiway cuts in node weighted graphs. Journal of Algorithms 50 (2004) 49–61.

[82] A.M.H. Gerards et B. Shepherd. Preselecting homotopies for the weighted disjoint paths problem. Manuscrit (1993).
Page web : http://cm.bell-labs.com/who/bshep/PS/homotopy.ps

[83] O. Goldschmidt et D.S. Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. Mathematics of operations research 19 (1994).

[84] D. Golovin, V. Nagarajan et M. Singh. Approximating the k-multicut problem. Actes SODA (2006) 621–630.

[85] M. Grötschel, L. Lovász et A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1 (1981) 169–197.

[86] J. Guo et R. Niedermeier. Fixed-parameter tractability and data reduction for multicut in trees. Networks 46 (2005) 124–135.

[87] J. Guo, F. Hüffner, E. Kenar, R. Niedermeier et J. Uhlmann. Complexity and Exact Algorithms for Multicut. Actes SOFSEM, LNCS 3831 (2006) 137–147.

[88] A. Gupta. Improved results for directed multicut. Actes SODA (2003) 454–455.

[89] V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd et M. Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. Journal of Computer and System Sciences 67 (2003) 473–496. Version préliminaire : actes STOC (1999) 19–28.

[90] M.T. Hajiaghayi et F.T. Leighton. On the max-flow min-cut ratio for directed multicommodity flows. Rapport technique MIT-LCS-TR-910 (2003).

[91] D. Hartvigsen. The planar multiterminal cut problem. Discrete Applied Mathematics 85 (1998) 203–222.

[92] G. Herman et A. Kuba. *Discrete Tomography - Foundations, Algorithms, and Applications*. Birkhäuser (2004).

[93] D. Hochbaum (éd.). *Approximation algorithms for NP-hard problems*. PWS Publishing Company, Boston (1997).

[94] T.C. Hu. Multicommodity network flows. Oper. Res. 11 (1963) 344–360.

[95] ILOG, S.A. CPLEX 9.0 Documentation.
Page web : http://esra.univ-paris1.fr/cplex/html/index.html

[96] T. Johnson, N. Robertson, P. Seymour et R. Thomas. Directed Tree-width. Journal of Combinatorial Theory, Series B 82 (2001) 138–154.

[97] D. Karger, P. Klein, M. Thorup, C. Stein et N. Young. Better Rounding Algorithms for a Geometric Embedding Relaxation of Minimum Multiway Cut. Actes STOC (1999).

[98] R.M. Karp. Reducibility among combinatorial problems. Complexity of Computer Computations. R.E. Miller, J.W. Thatcher (éds). Plenum Press, New York (1972) 85–103.

[99] J.C.M. Keijsper, R.A. Pendavingh et L. Stougie. A linear programming formulation of Mader's edge-disjoint paths problem. Journal of Combinatorial Theory, Series B 96 (2006) 159–163.

[100] B.W. Kernighan et D.M. Ritchie. *Le langage C - Norme ANSI (2e édition)*. Dunod (2004).

[101] P. Klein, S. Plotkin, S. Rao et É. Tardos. Approximation Algorithms for Steiner and Directed Multicuts. Journal of Algorithms 22 (1997) 241–269.

[102] J. Kleinberg et É. Tardos. Disjoint paths in densely embedded graphs. Actes FOCS (1995) 52–61.

[103] J. Kleinberg. Approximation algorithms for disjoint paths problems. PhD thesis, MIT, Cambridge, MA (1996).

[104] J. Kleinberg et É. Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. Journal of Computer and System Sciences 57 (1998) 61–73.

[105] J. Kleinberg. An approximation algorithm for the disjoint paths problem in even-degree planar graphs. Actes FOCS (2005).

[106] S.G. Kolliopoulos et C. Stein. Approximation Algorithms for Single-Source Unsplittable Flow. SIAM J. Computing 31 (2002) 919–946.

[107] S.G. Kolliopoulos et C. Stein. Approximating Disjoint-Path Problems using Packing Integer Programs. Mathematical Programming, series A (99) 63–87 (2004). Version préliminaire : actes IPCO (1998).

[108] P. Kolman et C. Scheideler. Improved bounds for the unsplittable flow problem. Actes SODA (2002).

[109] P. Kolman. A note on the greedy algorithm for the unsplittable flow problem. Information Processing Letters 88 (2003) 101–105.

[110] E. Korach et M. Penn. Tight integral duality gap in the Chinese postman problem. Mathematical Programming 55 (1992) 183–191.

[111] E. Korach et M. Penn. A fast algorithm for maximum integral two-commodity flow in planar graphs. Discrete Applied Mathematics 47 (1993) 77–83.

[112] B. Korte, L. Lovász, H.J. Prömel et A. Schrijver (éds). *Paths, Flows and VLSI-Layout.* Algorithms and combinatorics 9 (1990). Springer-Verlag. Berlin.

[113] Y. Kortsarts, G. Kortsarz et Z. Nutov. Greedy approximation algorithms for directed multicuts. Networks 45 (2005) 214–217.

[114] M.E. Kramer et J. van Leeuwen. The complexity of wire routing and finding the minimum area layouts for arbitrary VLSI circuits. Advances in computing research 2: VLSI theory, F.P. Preparata (éd.), JAI Press, London (1984) 129–146.

[115] J.B. Kruskal. On the shortest spanning subtree of a graph and traveling salesman problem. Proc. Amer. Math. Soc. 7 (1956) 48–50.

[116] P. Kubat, A. Shulman, R. Vachani et J. Ward. Multicommodity flows in ring networks. INFORMS Journal on Computing 8 (1996) 235–242.

[117] F.T. Leighton et S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. J. ACM 46 (1999) 787–832.

[118] T. Lengauer. Combinatorial algorithms for integrated circuit layout. Teubner, Stuttgart (1990), 697 p. Wiley.

[119] L. Létocart et F. Roupin. A semidefinite approach to solve multicut in trees. Actes J. Opt. (2002).

[120] L. Létocart. Problèmes de multicoupes minimales et de multiflots maximaux en nombres entiers. Thèse de doctorat en informatique, CEDRIC-CNAM (2002).

[121] A. Levin et D. Segev. Partial Multicuts in Trees. Actes WAOA, LNCS 3879 (2006) 320–333.

[122] M.S. Levine. Fast randomized algorithms for computing minimum {3,4,5,6}-way cuts. Actes SODA (2000) 735–742.

[123] L. Liu et P.J. Wan. Maximal throughput in wavelength-routed optical networks. Dans : "Multichannel Optical Networks: Theory and Practice", volume 46 des DIMACS Series in Discrete Mathematics and Theoretical Computer Science (1998, AMS) 15–26.

[124] A. Löbel. Vehicle scheduling in public transit and lagrangian pricing. ZIB Berlin (1997). Preprint SC 97–16.

[125] D. Marx. Eulerian disjoint paths problem in grid graphs is NP-complete. Discrete Applied Mathematics 143 (2004) 336–341.

[126] D. Marx. Parameterized graph separation problems. Theoretical Computer Science 351 (2006) 394–406.

[127] G. Mermoud, M.A. Schaub et G. Théoduloz. Partitioning CiteSeer's Citation Graph. Rapport EPFL (2005).

[128] M. Middendorf et F. Pfeiffer. On the complexity of the disjoint paths problem. Combinatorica 13 (1993) 97–107.

[129] R. Möhring, D. Wagner et F. Wagner. VLSI network design: a survey. M.O. Ball, T.L. Magnanti, C.L. Monma, et G.L. Nemhauser (éditeurs), Handbooks in Operations Research/Management Science, Volume on Networks (1995). Elsevier.

[130] J. Naor et L. Zosin. A 2-approximation algorithm for the directed multiway cut problem. SIAM J. Computing 31 (2001) 477–482.

[131] T. Nguyen. On the Disjoint Paths Problem. À paraître dans Operations Research Letters (2006).

[132] T. Nishizeki, J. Vygen et X. Zhou. The edge-disjoint paths problem is NP-complete for series-parallel graphs. Discrete Applied Mathematics 115 (2001) 177–186.

[133] K. Obata. Approximate max-integral-flow/min-multicut theorems. Actes STOC (2004).

[134] H. Okamura et P.D. Seymour. Multicommodity flows in planar graphs. Journal of Combinatorial Theory, Series B 31 (1981) 75–81.

[135] C. Papadimitriou et M. Yannakakis. Optimization, approximation, and complexity classes. J. Comput. and System Sciences 43 (1991) 425–440.

[136] V. Th. Paschos. *Complexité et approximation polynomiale.* Hermès Science, Paris (2004).

[137] D. Peleg et E. Upfal. Disjoint paths on expander graphs. Combinatorica 9 (1989) 289–313.

[138] J.S. Provan et R.C. Burk. Two-Connected Augmentation Problems in Planar Graphs. Journal of Algorithms 32 (1999) 87–107.

[139] Y. Rabani. Path coloring on the mesh. Actes FOCS (1996) 400–410.

[140] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. Journal of Computer and System Sciences 37 (1988) 130–143.

[141] P. Raghavan et E. Upfal. Efficient routing in all optical networks. Actes STOC (1994) 134–143.

[142] S. Rajagopalan. Two commodity flows. Oper. Res. Letters 15 (1994) 151–156.

[143] S. Rao et S. Zhou. Edge Disjoint Paths in Moderately Connected Graphs. Actes ICALP (2006) 202–213.

[144] G. Rinaldi. RUDY: a generator for random graphs. IASI Rome (1996). Page web : http://www-user.tu-chemnitz.de/∼helmberg/sdp_software.html

[145] N. Robertson et P.D. Seymour. Graph minors II: Algorithmic aspects of tree-width. Journal of Algorithms 7 (1986) 309–322.

[146] N. Robertson et P.D. Seymour. Graphs minors XIII: The disjoint paths problem. J. Comb. Theory, Ser. B, 63 (1995) 65–110.

[147] F. Roupin. From Linear to Semidefinite Programming: an Algorithm to obtain Semidefinite Relaxations for Bivalent Quadratic Problems. Journal of Combinatorial Optimization 8 (2004) 469–493.

[148] M. Saks, A. Samorodnitsky et L. Zosin. A Lower Bound On The Integrality Gap For Minimum Multicut In Directed Networks. Combinatorica 24 (2004) 525–530.

[149] A. Schrijver. *Theory of linear and integer programming.* Interscience series in discrete mathematics and optimization (1986). Wiley.

[150] A. Schrijver, P. Seymour et P. Winkler. The Ring Loading Problem. SIAM Journal on Discrete Mathematics 11 (1998) 1–14.

[151] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency.* Algorithms and Combinatorics 24 (2003). Springer.

[152] A. Sebö. Integer plane multicommodity with a fixed number of demands. J. Combin. Theory, Ser. B, 59 (1993) 163–171.

[153] P. Seymour. Matroids and multicommodity flows. European Journal of Combinatorics 2 (1981) 257–290.

[154] A. Slivkins. Parameterized Tractability of Edge-Disjoint Paths on Directed Acyclic Graphs. Actes ESA (2003).

[155] A. Srinivasan. Approximation algorithms via randomized rounding: a survey. Rapport technique Bell Lab. (2000).

[156] A. Srivastav et P. Stangier. On complexity, representation and approximation of integral multicommodity flows. Discrete Applied Mathematics 99 (2000) 183–208.

[157] É. Tardos et V.V. Vazirani. Improved bounds for the max-flow min-multicut ratio for planar and $K_{r,r}$-free graphs. Inform. Proc. Lett. 47 (1993) 77–80.

[158] K.R. Varadarajan et G. Venkataraman. Graph decomposition and a greedy algorithm for edge-disjoint paths. Actes SODA (2004) 379–380.

[159] V.V. Vazirani. *Approximation algorithms.* Springer-Verlag (2001).

[160] J. Vygen. Disjoint paths. Rapport technique 94816, Research Institute for Discrete Mathematics, University of Bonn (1994).

[161] J. Vygen. NP-completeness of some edge-disjoint paths problem. Discrete Applied Mathematics 61 (1995) 83–90.

[162] K. Weihe. Edge-disjoint routing in plane switch graphs in linear time. Actes FOCS (1999) 330–340. New York City.

[163] M. Yannakakis, P. Kanellakis, S. Cosmadakis et C. Papadimitriou. Cutting and partitioning a graph after a fixed pattern. Actes ICALP, Lecture Notes in Computer Science 154 (1983) 712–722.

[164] W.-C. Yeh. A simple algorithm for the planar multiway cut problem. J. Algorithms 39 (2001) 68–77.

[165] M. Zrikem. Optimisation du routage de câbles dans les installations de production d'électricité. Thèse de doctorat en informatique, CEDRIC-CNAM (2001).

Annexes

Annexe A

Travaux complémentaires

Durant ma dernière année de thèse, et parallèlement à mes autres travaux de recherche, j'ai eu l'occasion de travailler (en collaboration avec Marie-Christine Costa et Christophe Picouleau) avec Bernard Ries et le professeur Dominique de Werra de l'EPFL (Lausanne) sur des problèmes de reconstruction de graphes colorés, qui généralisent les problèmes classiques issus de la tomographie discrète [92]. Ces travaux n'ont pas été inclus dans cette thèse, en particulier pour des raisons d'homogénéité, mais nous les décrivons brièvement dans cette annexe.

Le premier problème que nous avons étudié est le suivant : étant donnés un graphe, une collection de chaînes de ce graphe, et, pour chaque chaîne, le nombre de sommets de chaque couleur, on cherche à colorer les sommets du graphe de façon à respecter les projections de couleurs données sur l'ensemble des chaînes. Nous avons obtenu plusieurs résultats de complexité intéressants pour ce problème ; en particulier, pour deux couleurs, il est $\mathcal{NP}$-complet dans les arbres de degré maximum 3 et dans les graphes complets. Nous avons également exhibé un certain nombre de cas polynomiaux : par exemple, ce problème est polynomial dans les arbres si deux chaînes quelconques ont au plus un sommet en commun. L'ensemble de ces résultats a fait l'objet d'un article actuellement en soumission. Une version préliminaire est disponible à l'adresse suivante :

http://cedric.cnam.fr/∼bentz/these/colorationChaines.ps

Le deuxième problème étudié consistait à reconstruire un graphe dont les sommets sont colorés, uniquement à partir de la liste des degrés colorés des sommets : en d'autres termes, pour chaque sommet, on connaît le nombre de voisins de chaque couleur (remarquons que, dans ce problème, le graphe n'est donc pas donné a priori). Nous avons conçu des algorithmes polynomiaux pour différentes topologies de graphes (c'est-à-dire, en imposant que le graphe que l'on reconstruit doit être une chaîne ou un arbre, par exemple), le cas général étant $\mathcal{NP}$-complet. Un article regroupant l'ensemble de ces résultats est actuellement en préparation.

Enfin, j'ai eu l'occasion de travailler avec Christophe Picouleau sur des problèmes de coloration bornée dans des arbres. Nous avons conçu un algorithme de programmation dynamique (polynomial si le nombre de couleurs est fixé) résolvant tout problème de coloration de sommets dont la fonction objectif (i) est calculable en temps polynomial en tout point donné et (ii) ne dépend que des cardinalités des différentes classes de couleur. Cette collaboration a fait l'objet d'un article actuellement en soumission. Une version préliminaire est disponible à l'adresse suivante :

http://cedric.cnam.fr/∼bentz/these/colorationBorneeArbres.pdf

Annexe B

Glossaire

B.1 Liste des acronymes associés aux problèmes étudiés

- CDA : problème de l'existence de Chemins Disjoints par les Arêtes ;
- CDS : problème de l'existence de Chemins Disjoints par les Sommets ;
- CMEM : problème de la Coupe Multi-Ensemble Minimum ;
- CMTM : problème de la Coupe MultiTerminale Minimum ;
- CMTMS : problème de la Coupe MultiTerminale Minimum sur les Sommets ;
- FMTEM : problème du Flot MultiTerminal Entier Maximum ;
- KC : problème de la K-Coupe ;
- MCDA : problème de la Maximisation du nombre de Chemins Disjoints par les Arêtes ;
- MCDS : problème de la Maximisation du nombre de Chemins Disjoints par les Sommets ;
- MCLM : problème de MultiChemins à Longueur Minimum ;
- MCM : problème de la MultiCoupe Minimum ;
- MCM-FC : variante du problème de la MultiCoupe Minimum où l'on cherche à faire disparaître les composantes Fortement Connexes ;
- MCMS : problème de la MultiCoupe Minimum sur les Sommets ;
- MFCD : problème du MultiFlot Continu avec Demandes ;
- MFECM : problème du MultiFlot Entier Concurrent Maximum ;
- MFED : problème du MultiFlot Entier avec Demandes ;
- MFEM : problème du MultiFlot Entier Maximum ;
- MFEMD : problème du MultiFlot Entier avec le Maximum de Demandes ;
- MFEMI : problème du MultiFlot Entier Maximum Insécable ;
- MLCDA : problème de la Maximisation du nombre de Liaisons routées suivant des Chemins Disjoints par les Arêtes.

B.2 Liste des abréviations utilisées dans ce document

- GSC : Graphe(s) orienté(s) Sans Circuits ;
- PL : Programme(s) Linéaire(s) ou Programmation Linéaire ;
- PLNE : Programme(s) Linéaire(s) en Nombres Entiers ou Programmation Linéaire en Nombres Entiers ;
- TU : Totalement Unimodulaire(s) (pour une ou des matrice(s)).

Annexe C

Bilan et problèmes ouverts

C.1 Tableaux récapitulatifs

Les résultats obtenus dans cette thèse sont indiqués en gras et en italique.

Graphes	MFEM	MCM	MFEMI	MCDA	CMTM	FMTEM
Orientés	$m^{\frac{1}{2}-\varepsilon}$–inapprox. [89] $O\left(\sqrt{m}\right)$–approx. [103]	APX–diff. pour $\|\mathcal{L}\| = 2$ [78] $O(\sqrt{n})$–approx. [88]	$\mathcal{NP}$-diff. si $\|\mathcal{L}\| = 2$ [67] $O\left(\sqrt{m}\right)$–approx. [103]	Idem MFEM	APX-diff. si $\|\mathcal{L}\| = 2$ [78] 2–approx. [130]	***(2-ε)–inapprox., même pour $\|\mathcal{L}\| = 2$*** ***$2\log_2(k_S+2)$–approx.***
GSC	$\mathcal{NP}$-diff. si $\|\mathcal{L}\| = 2$ [64] $O\left(\sqrt{n}\right)$–approx. [39]	***$\mathcal{NP}$-diff. si degrés bornés et poids uniformes***		Idem MFEM	Polyn. [52]	Polyn. [52]
De largeur d'arbre orientée bornée		***APX-diff. même si degrés bornés et poids uniformes***				
Anneaux	***Polyn.***	Polyn. [52]	***Polyn.***	Polyn. [123]	Polyn. [52]	Polyn. [52]
Arbres bi-orientés	APX-diff. [62]		APX-diff. [62]	APX-diff. $(\frac{5}{3}+\varepsilon)$–approx. [62]		
Arbres orientés	Polyn. [52]	Polyn. [52]	Polyn. [52]	Polyn. [52]	Polyn. [52]	Polyn. [52]

Tab. C.1 – Résultats principaux pour MFEM, MCM et leurs variantes (graphes orientés).

Graphes	MFEM	MCM	MFEMI	MCDA	CMTM	FMTEM
Non orientés	$\mathcal{NP}$-diff. si $\|\mathcal{L}\| = 2$ [64] $(\log m)^{\frac{1}{2}-\varepsilon}$– inapprox. [7, 8] $O(1)$–approx. si capacités sont $\Omega(\log n)$ [140]	APX-diff. pour $\|\mathcal{L}\| = 3$ [54] $O(\log \|\mathcal{L}\|)$–approx. [79] Pas dans APX si Unique Games Conjecture est vraie [31]	$(\log m)^{\frac{1}{2}-\varepsilon}$– inapprox. [7, 8] $O\left(\sqrt{n}\right)$–approx. [39]	$\mathcal{NP}$-diff. si $\|\mathcal{L}\| = 2$ [64] $(\log m)^{\frac{1}{2}-\varepsilon}$– inapprox. [7, 8] $O\left(\sqrt{n}\right)$–approx. [39]	APX-diff. pour $\|\mathcal{L}\| = 3$ [54] 1.3438–approx. [97]	Polyn. [99]
Planaires	APX-diff. [80] $O(\log n)/O(1)$–approx. si capacités $\geq 2/4$ [35, 37, 40] $O(\log^2 n)$–approx. si degrés pairs [105]	APX-diff. [80] $O(1)$–approx. [157] ***Polyn. si $\|\mathcal{L}\|$ fixé et terminaux sur la face extérieure***	APX-diff. [80]	APX-diff. [80] ***4k–approx. dans graphes planaires à k niveaux d'arêtes***	$\mathcal{NP}$-diff. Polyn. pour $\|\mathcal{L}\|$ fixé [54] Polyn. si terminaux sur la face extérieure [42]	
Arbres	APX-diff. 2–approx. [80]	APX-diff. 2–approx. [80]	APX-diff. 2–approx. [80]	Polyn. [80]	Polyn. $O(n)$ [51]	Polyn. $O\left(n^2\right)$ [51]
De nb. cyclomatique $\gamma \geq 0$	APX-diff. [80] ***2(γ+1)–approx.***	APX-diff. [80] ***2(γ+1)–approx.***	APX-diff. [80] ***2(γ+1)–approx.***	***Polyn. pour γ fixé***		
De largeur d'arbre bornée	APX-diff. [80] $O(\log n)$–approx. [41]	APX-diff. [80] PTAS si degrés bornés et poids uniformes [27] ***Polyn. si $\|\mathcal{L}\|$ est fixé***	APX-diff. [80]	APX-diff. [80]	Polyn. [54]	
Anneaux	***Polyn.***	Polyn. [52]		Polyn. [123]	Polyn. [52]	Polyn. [52]
Arbres d'anneaux	APX-diff. [80] ***4–approx.***	APX-diff. [80] ***4–approx.***	APX-diff. [80] ***4–approx.***	APX-diff. [80] ***4–approx.*** (3–approx. pour MLCDA [61])	Polyn. [42]	
Grilles uniformes	$\mathcal{NP}$-diff. [114]	***$\mathcal{NP}$-diff.***	$\mathcal{NP}$-diff. [114]	$\mathcal{NP}$-diff. [114] $O(1)$–approx. [102]		

TAB. C.2 – Résultats principaux pour MFEM, MCM et leurs variantes (graphes non orientés).

C.2 10 problèmes ouverts

1. Quelle est la complexité de MCDA dans les graphes planaires (orientés ou non) si le nombre de liaisons est fixé ?
2. Quelle est la complexité de MCDA dans les graphes (orientés ou non) de largeur d'arbre bornée si le nombre de liaisons est fixé ?
3. Quelle est la complexité de MCDA dans les graphes de largeur d'arbre bornée si le degré maximum est borné ?
4. Existe-t-il un algorithme $O(1)$-approché pour MFEM dans les graphes de largeur d'arbre bornée ?
5. Peut-on améliorer l'approximation de MCDA dans les graphes non orientés ? Et dans les graphes planaires ?
6. Existe-t-il un algorithme $O(1)$-approché pour FMTEM dans les graphes orientés ?
7. Quelle est la complexité de CMTM dans les graphes planaires si les terminaux sont situés sur un nombre fixé de faces ?
8. Existe-t-il un PTAS pour CMTM dans les graphes planaires ?
9. CMTM est-il FPT dans les graphes planaires (si le nombre de terminaux est vu comme le paramètre) ?
10. Quelle est la complexité de MCM si le nombre de liaisons est fixé et si le graphe est un GSC ? Si le graphe est non orienté et planaire ?

Résumé

Titre : *Résolution exacte et approchée de problèmes de multiflot entier et de multicoupe : algorithmes et complexité.*

Dans cette thèse, on s'intéresse à des problèmes de multiflot entier et de multicoupe, qui généralisent les problèmes classiques de flot maximum et de coupe minimum. Deux aspects de ces problèmes y sont étudiés en particulier : la résolution exacte en temps polynomial et l'approximation polynomiale.

Du point de vue de la résolution exacte, nos principales contributions portent sur les sujets suivants :
- Chemins disjoints dans les graphes de nombre cyclomatique borné ;
- Multicoupes dans les graphes orientés sans circuits, dans les graphes non orientés de largeur d'arbre bornée et dans les graphes planaires ;
- Multiflots entiers dans les anneaux ;
- Multicoupes et multiflots entiers dans certains cas particuliers de grilles.

Du point de vue de l'approximation polynomiale, nos principales contributions portent sur les sujets suivants :
- Chemins disjoints dans les graphes planaires à k niveaux d'arêtes ;
- Multiflots entiers dans les graphes de nombre cyclomatique borné ;
- Multicoupes dans les graphes orientés non pondérés de largeur d'arbre et de degré maximum bornés ;
- Flots multiterminaux entiers dans les graphes orientés.

Nous décrivons également une nouvelle heuristique pour trouver un multiflot entier maximum dans un graphe non orienté, et la testons sur des instances générées aléatoirement.

Mots-clés : *Optimisation combinatoire, Optimisation dans les graphes, Multiflots entiers, Multicoupes, Chemins disjoints, Algorithmes polynomiaux, Approximation polynomiale, $\mathcal{NP}$-difficulté, APX-difficulté, Planarité, Largeur d'arbre.*

Abstract

Title: *Exact and approximate resolution of integral multiflow and multicut problems: algorithms and complexity.*

In this thesis, we consider integral multiflow and multicut problems, which generalize the classical max flow and min cut problems. Two aspects of these problems are studied in particular: polynomial-time resolution and polynomial approximation.

Concerning the first aspect, our main contributions focus on the following points:
- Disjoint paths in graphs of bounded cyclomatic number;
- Multicuts in directed acyclic graphs, in undirected graphs of bounded tree-width and in planar graphs;
- Integral multiflows in rings;
- Multicuts and integral multiflows in several special types of grids.

Concerning the second aspect, our main contributions focus on the following points:
- Disjoint paths in k-edge-outerplanar graphs;
- Integral multiflows in graphs of bounded cyclomatic number;
- Multicuts in unweighted digraphs of bounded maximum degree and bounded tree-width;
- Integral multiterminal flows in digraphs.

We also describe a new heuristic to find a maximum integral multiflow in an undirected graph, and test it on randomly generated graphs.

Keywords: *Discrete optimisation, Graph optimisation, Integral multiflows, Multicuts, Disjoint paths, Polynomial algorithms, Polynomial approximation, $\mathcal{NP}$-hardness, APX-hardness, Planarity, Tree-width.*