

U.F.R SCIENCES DE LUMINY

ÉCOLE DOCTORALE DE MATHÉMATIQUE ET INFORMATIQUE, E.D 184

Thèse de DoctoratFLOTS ET CHEMINS CONTRAINTS : APPLICATIONS AUX RÉSEAUX DE
TÉLÉCOMMUNICATIONS**Sébastien Imbrosciano**

En vue d'obtenir le grade de Docteur de l'Université AIX-MARSEILLE

Spécialité : Informatique**Soutenance à la date de 21/01/2014**

Rapporteur : Marie-Christine COSTA - ENSTA-ParisTech
Philippe MAHEY - ISIMA Université Blaise Pascal

Directeurs : Alexandre LAUGIER - France Télécom
Yann VAXÈS - Université Aix-Marseille

Examineurs : Stéphane BESSY - LIRMM Université Montpellier II
Victor CHEPOI - Université Aix-Marseille
Jérôme GALTIER - France Télécom

Laboratoire d'Informatique Fondamentale de Marseille - **CNRS UMR 7279**
Orange Labs R&D - **France Telecom**



Remerciements

À faire en dernier

Table des matières

1	Introduction	1
1.1	Contexte technologique	1
1.2	Problématiques	3
1.3	Contributions	4
2	Préliminaires	7
2.1	Terminologies des graphes	7
2.2	Complexité et algorithmique	10
2.3	Programmation Linéaire	17
2.4	Problème de flots et variantes	25
2.5	Problème de plus court chemins et variantes	29
3	Problème de déploiement FTTH	41
3.1	Contexte du problème FTTH	41
3.2	Modélisation du problème FTTH sous architecture PON	46
3.3	Résolution de la relaxation linéaire	53
3.4	Heuristique de génération d'une solution entière	59
3.5	Résultats des simulations	61
3.6	Conclusions	66
4	Problème de délai dans la conception de réseaux	67
4.1	Contexte du problème de délais	67
4.2	Problème de flot avec contrainte de délai convexe	71
4.3	Problème de flot avec contrainte de délai proportionnel	75
4.4	Algorithme d'approximation facteur 2 pour le graphe chemin	79
4.5	Modélisation en un programme linéaire disjonctif	83
4.6	Heuristique par recherche locale	87

4.7	Conclusions	92
5	Protocoles pour les réseaux de capteurs	95
5.1	Contexte	96
5.2	Routage sûr de l'information dans les réseaux de capteurs	99
5.3	Gestion de l'énergie dans les réseaux de capteurs	118
5.4	Conclusions	125
6	Conclusion	127
	Bibliographie	131
	Annexes	145
	Annexe 1 : Brevet	147
	Annexe 2 : Algotel2011 - FTTH	181

Introduction

1

1.1 Contexte technologique

La télécommunication est au centre de notre mode de vie actuel. Elle peut être définie comme la communication à distance par des moyens technologiques. Les technologies de télécommunication électriques et électromagnétiques ont subi de grandes évolutions en commençant par l'invention du télégraphe, suivi du téléphone, qui a permis plus tard l'accès à tous les particuliers à Internet. De nos jours, l'évolution se porte sur l'accès pour tous à la fibre optique. En parallèle des technologies de communications sans fils se sont développées avec l'utilisation de faisceaux hertziens qui permettent la transition d'informations sur de longues distances comme la radio, la télévision, la téléphonie mobile et les communications par satellites.

Un réseau de télécommunications est un ensemble de nœuds terminaux, de liens et de nœuds intermédiaires qui sont connectés de manière à permettre la télécommunication entre les terminaux. Chaque terminal dans le réseau possède généralement une adresse unique afin que les messages ou les connexions puissent être acheminés vers les destinataires appropriés. Un réseau informatique est un réseau de télécommunications qui relie un ensemble d'ordinateurs pour permettre la communication et l'échange de données entre les systèmes, les applications logicielles et les utilisateurs. Les protocoles de communication définissent des règles (normes) et des formats de données pour l'échange d'informations dans un réseau informatique. Le réseau informatique le plus connu est Internet.

Internet. Le réseau Internet est un système mondial de réseaux informatiques qui utilisent la suite de protocoles Internet standard TCP/IP pour servir des milliards d'utilisateurs à travers le monde. C'est un réseau de réseaux qui se compose de millions de réseaux privés, publics, universitaires, entre autres. Par exemple la

figure 1.1 représente le réseau Internet mère en Europe modélisé par un graphe. Les sommets correspondent aux points de raccordement du réseau, et une arête relie deux sommets s'il existe un câble réseau reliant les deux points de raccordement correspondants. À chaque arête est associée un entier correspondant au débit théorique d'information pouvant circuler entre les deux sommets.

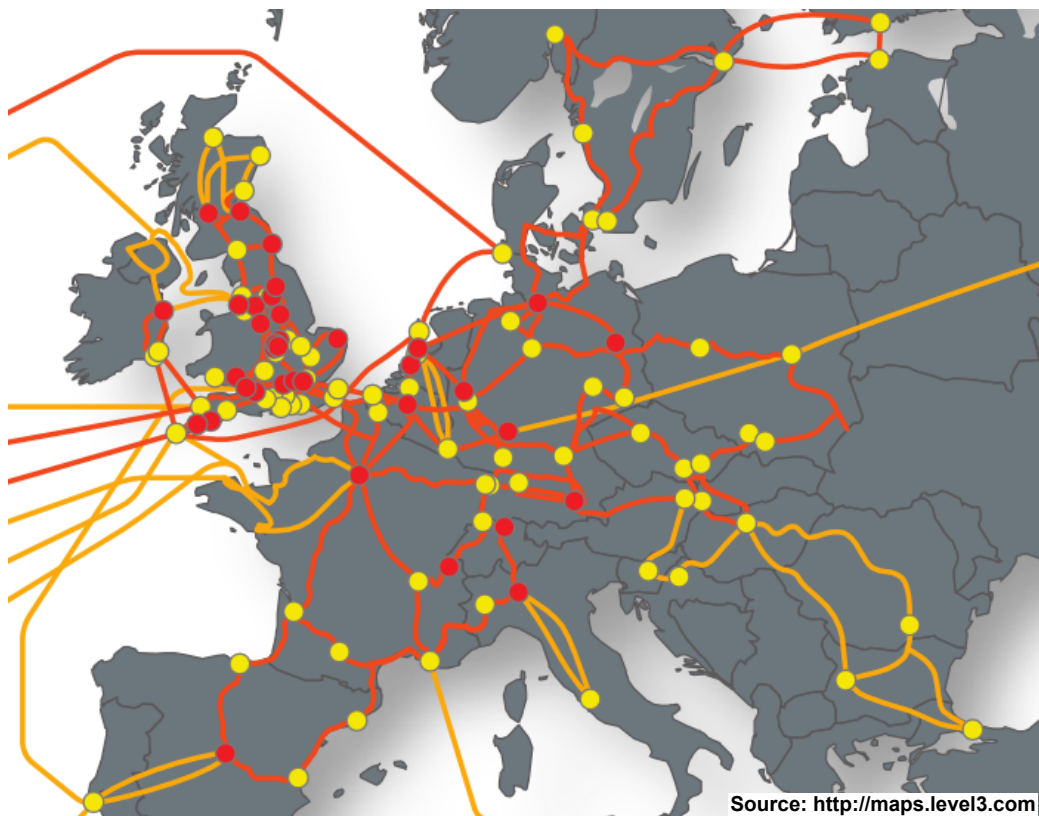


FIGURE 1.1 – Graphe du réseau Internet mère en Europe.

Internet comporte un vaste éventail de ressources et de services d'information notamment les documents hypertextes liés entre eux via le “World Wide Web” (WWW). Les moyens de communication traditionnels tels que le téléphone, la musique, le cinéma ou la télévision ont été redéfinis par Internet. Cela a donné naissance à de nouveaux services tels que la voix sur protocole Internet (VoIP) et la télévision sur protocole Internet (IPTV). Les journaux, livres et autres publications imprimées s'adaptent à la technologie du site Web, où fleurissent blogs et flux RSS. Internet accélère et crée de nouvelles formes d'interactions humaines grâce à la messagerie instantanée, les forums Internet et les réseaux sociaux (Internet/web 2.0). Le volume des achats en ligne explose pour les grands magasins mais aussi pour les petits

artisans et commerçants. Des services Business-to-Business et financiers sur Internet modifient en profondeur les chaînes d'approvisionnement des industries. Cela fait d'Internet un acteur indissociable de l'économie mondiale. Pourtant à l'origine en 1960, Internet était un réseau informatique du gouvernement des États-Unis qui avait pour but de faciliter la communication entre chercheurs. Or il n'a cessé de grandir jusqu'à devenir international. Dans les années 1990, apparaissent les premières pages Web et Fournisseurs d'Accès à Internet (*F.A.I*) privatisant ainsi les équipements du réseau.

Réseaux de capteurs. Le point de départ de la recherche sur les capteurs, dans les années 1990, fût la mise en production de minuscules capteurs à faibles coûts, grâce à la miniaturisation des composants électroniques. Les capteurs sont de véritables systèmes embarqués dont les principaux éléments sont un micro-contrôleur, un émetteur-récepteur, une mémoire externe, la source d'alimentation et un ou plusieurs transducteurs. Le micro-contrôleur effectue les tâches/processus et commande le fonctionnement des autres composants du capteur. L'émetteur-récepteur s'occupe de la télécommunication. La mémoire externe stocke les données du transducteur. Dans la plupart des cas, une batterie et le transducteur convertit un signal physique en un signal électrique transmis au micro-contrôleur.

Un réseau de capteurs sans fil est constitué de capteurs autonomes répartis dans l'espace pour surveiller les conditions physiques ou environnementales telles que la température, le son, la pression, entre autres. Ensuite ces données sont transmises via leur propre système de télécommunication. Le développement des réseaux de capteurs sans fil a été motivé par les applications militaires telles que la surveillance de champ de bataille. Aujourd'hui ces réseaux sont utilisés dans de nombreuses applications industrielles et grand public, telles que la surveillance de processus industriels et de contrôle, la surveillance environnementale, le contrôle du trafic routier urbain, la prévention d'accidents ou encore la domotiques.

1.2 Problématiques

Problématique des F.A.I dû aux nouveaux usages d'Internet. Une problématique centrale dans ce type de réseaux est de satisfaire les besoins de débits élevés symétriques avec un besoin de recevoir mais aussi d'envoyer des données avec un temps de réponse court qui se traduit par l'augmentation forte du volume des informations échangées. Pour cela, les *F.A.I* orientent progressivement le grand public vers le très haut débit. Cette nouvelle offre de service grand public implique le

déploiement de réseaux de fibres optiques jusqu'à l'abonné, qui, jusque là, n'étaient utilisés que pour le réseau mère du *F.A.I.* Celui-ci se reposait sur les réseaux téléphoniques pré-existants pour transmettre les informations jusqu'au domicile du consommateur.

Le visionnage de vidéo par Internet ou les nouveaux usages des entreprises tel que la dématérialisation totale (Cloud-computing) impose aux *F.A.I.* des contraintes de qualité de service qui se traduisent par une prise en compte plus fine du délai de transmission des informations dans son réseau mère.

Problématique des nouveaux usages des capteurs. L'étude des réseaux de capteurs sans fil constitue un domaine de recherche actif avec de nombreux ateliers et conférences organisés chaque année pour répondre aux problèmes de ce type de réseaux. La gestion de la consommation de la batterie d'alimentation du capteur et la prise en compte de la topologie variable soumise aux aléas de son environnement représentent deux axes principaux de recherche de ce domaine.

1.3 Contributions

Problème de déploiement FTTH. Dans le Chapitre 3, nous avons modélisé le problème de déploiement de réseaux de fibres optiques jusqu'au domicile de l'abonné [92]. Ce problème consiste à concevoir un plan d'installation de fibres optiques de coût minimal permettant de connecter un ensemble de clients à un nœud de raccordement via un ensemble de coupleurs en respectant les contraintes techniques imposées par la norme. Cette modélisation se base sur la notion de flot avec multiplicateurs. Nous avons défini le programme linéaire correspondant à notre problème pour cette modélisation. Nous avons montré comment obtenir par génération de colonnes une solution optimale fractionnaire du programme linéaire. Cette approche génère un sous-problème qui est un problème de plus court chemin généralisé multi-contraint. Nous avons résolu par un algorithme pseudo-polynomial qui généralise au cas avec multiplicateurs l'algorithme de plus court chemin multi-contraint. Pour finir, nous avons détaillé une méthode d'arrondis de la solution fractionnaire, ainsi que des méthodes augmentant la convergence de la solution fractionnaire en mettant à profit des données calculées par l'algorithme de plus court chemin. Nous sommes parvenus à obtenir des solutions de bonne qualité en temps raisonnable. Les simulations effectuées nous permettent de souligner l'importance de la prise en charge des informations calculées lors de la résolution de l'algorithme de plus court chemin pour améliorer la convergence de la procédure de génération de colonnes.

Problème de délai dans la conception de réseaux. Dans le Chapitre 4, nous nous sommes intéressés aux problèmes d’optimisation liés aux calculs de routages dans des réseaux de communications. Ces problèmes sont généralement modélisés grâce à la notion classique de flot ou de multiflot dans un graphe. L’étude de la *Qualité de Service* dans les réseaux de télécommunications conduit à prendre en compte le délai d’acheminement des messages à travers le réseau. Nous devons introduire une contrainte supplémentaire qui impose que la somme des délais sur les arêtes d’un chemin de routage, i.e. un chemin qui porte une quantité non nulle de flot, doit être bornée par une constante donnée. Lorsque le délai pour traverser une liaison dépend du taux de congestion de celle-ci, la théorie des files d’attente et les expérimentations conduisent à utiliser des fonctions convexes pour modéliser le délai de transmission. Le problème devient beaucoup plus difficile à résoudre ou à approximer.

Nous avons étudié le cas où la fonction de délai est proportionnelle. Nous avons proposé deux preuves de NP-complétude, portant sur deux réductions de notre problème. Nous avons présenté ensuite un algorithme d’approximation facteur 2 sur le graphe chemin avec demandes uniformes. Pour finir, nous avons proposé des heuristiques efficaces capables de calculer en un temps raisonnable de bonnes solutions pour des instances de tailles réelles.

Protocoles pour les réseaux de capteurs. Le travail du Chapitre 5 est le fruit de solutions apportées à deux problèmes d’optimisations rencontrés par une équipe de recherche travaillant sur des capteurs communicants dans le but de créer des villes connectées. Les capteurs qui sont des composants électroniques à faibles coûts peuvent être déployés partout pour mesurer et transmettre des informations sur leurs environnements. Le champ d’application des capteurs appartient à différents domaines tels que la sécurité, la météorologie, la surveillance de chaîne de support logistique.

Les capteurs de demain ne se cantonneront pas seulement à mesurer leurs environnements et à transmettre cette information. Ils seront aussi capables d’interroger directement d’autres capteurs du réseau pour associer leurs mesures et ainsi traiter l’information. Leur mode de transmission par voie radio électrique est soumis à des aléas. Les capteurs doivent pouvoir assurer eux même la topologie virtuelle du réseau. Pour éviter qu’une interruption de la communication sur un lien impacte fortement la transmission des données, il est souhaitable qu’elles transitent sur un ensemble de chemins disjoints, i.e. n’empruntant pas la même liaison radio entre deux capteurs. Nous avons proposé un protocole basé sur un algorithme distribué

permettant de maximiser le nombre de chemins disjoints entre un sous-ensemble de capteurs. Ce protocole de gestion de réseaux de capteurs a donné lieu à un brevet international [91].

Une des problématiques centrales dans le domaine des réseaux de capteurs est liée au fait que les capteurs sont alimentés par des batteries dont l'autonomie est limitée. Les protocoles de routage des communications à travers les réseaux de capteurs doivent prendre en compte cette limitation. Nous avons proposé un protocole qui permet de calculer un routage des communications de façon à maximiser la durée de vie du réseau de capteurs. Ce protocole fait l'objet d'un dépôt de brevet (voir Annexe 1).

Outils et techniques utilisés. Pour ces différents problèmes, nous avons utilisé les techniques du domaine de la recherche opérationnelle et plus particulièrement de l'optimisation combinatoire. Cela nous a amené à employer des outils de théorie des graphes, de complexité, d'algorithmique (distribuée) et de programmation linéaire.

Plus particulièrement, nous avons exprimé les problèmes rencontrés sous forme de problème d'optimisation. Dans la plupart des cas, nous avons défini un programme linéaire (en nombre entiers) pour modéliser le problème. Ces programmes linéaires correspondent à des variantes de problème de flot. Lorsque ce modèle comporte un grand nombre de variables, nous avons utilisé une méthode de génération de colonnes pour le résoudre. Cette méthode fait appel à un sous-problème de séparation qui, dans le cas des variantes des problèmes de flot, est une variante du problème de plus court chemin. Dans le Chapitre 3, nous avons résolu cette variante de plus court chemin par un algorithme de programmation dynamique. Dans le Chapitre 4, nous avons utilisé la programmation disjonctive. Pour ce problème, nous avons proposé un algorithme d'approximation facteur $\frac{1}{2}$ dans le graphe chemin pour des demandes uniformes qui utilise un algorithme polynomial pour calculer un stable de poids maximum dans les graphes parfaits.

Avant de présenter nos résultats, il convient de rappeler les principales définitions et notions usuelles qui seront utiles dans la suite de cette thèse. En particulier, nous insisterons sur les notions et algorithmes de flots, de plus court chemin et de génération de colonnes qui constituent la base de nos travaux et résultats. Ce chapitre nous permet également d'introduire le vocabulaire et les notations que nous utiliserons par la suite.

2.1 Terminologies des graphes

Cette section traite des principales définitions et notations relatives aux graphes. Ces notations se basent sur des ouvrages de référence sur la théorie des graphes, notamment “Graphes et Hypergraphes” [25] et “Graph Theory” [54].

Graphe non orienté

Un *graphe non orienté* G est composé d'un couple (V, E) , où V est un ensemble de *sommets*, et E un ensemble d'*arêtes* formées de paires de sommets distincts. Une arête entre les sommets u et v est notée $e = \{u, v\} \in E$. Nous disons qu'un sommet $u \in V$ est *incident* à e , s'il existe $e = \{u, v\} \in E$. Dans ce cas, le sommet u est une *extrémité* de e . Deux sommets $u, v \in V$ sont dits *adjacents*, ou *voisins*, s'il existe une arête $e = \{u, v\} \in E$. Deux arêtes $e_1, e_2 \in E$ sont dites *adjacentes* si elles partagent un sommet incident commun $v \in V$, i.e $e_1 = \{u, v\}$ et $e_2 = \{v, w\}$.

De manière usuelle, nous notons $V(G)$ et $E(G)$ l'ensemble des sommets et l'ensemble des arêtes du graphe G .

Graphe orienté

un graphe orienté G ou *digraphe* est un couple (V, E) , où V est un ensemble de sommets, et E un ensemble d'arcs. Un *arc* $e = (u, v) \in E$ est un couple de sommets, c'est à dire une arête orientée. Dans un digraphe, les arcs $e_1 = (u, v) \in E$ et $e_2 = (v, u) \in E$ sont distincts.

Topologie sur les graphes

Voici une liste non exhaustive de relations couramment utilisées sur les graphes que nous employons dans ce document. Le *voisinage* (ouvert) d'un sommet $v \in V$, noté $N_G(v)$, est l'ensemble des sommets adjacents à v , i.e $N(v) = \{u \mid \{u, v\} \in E\}$. On définit le *voisinage fermé* $N[v] = N(v) \cup v$. Le *degré* d'un sommet $v \in V$, noté $d_G(v)$, est le nombre de ses voisins, i.e $d_G(v) = |N_G(v)|$. Le sous-graphe G' de G induit par le sous-ensemble de sommets $S \subset V(G)$ et le graphe dont l'ensemble de sommets est S et l'ensemble des arêtes sont les arêtes de G dont les 2 extrémités sont dans S , i.e $E(G') = \{\{u, v\} \in E(G) \mid u, v \in V(G')\}$. On note graphe $G[S]$ le sous-graphe induit par l'ensemble de sommets $S \subseteq V(G)$ de G . Un graphe G est *connexe* s'il est possible de relier toutes paires de sommets en passant par les arêtes du graphe. Une *composante connexe* $C(G)$ de G est un sous-graphe connexe de G maximal par inclusion, i.e aucun autre sommet du graphe ne peut être ajouté à $C(G)$ sans que ce dernier ne devienne un sous-graphe non connexe. Une *clique* est un sous-graphe complet d'un graphe G dont la taille correspond au nombre de sommets qu'elle contient, une clique contenant n sommets est notée K_n . Le *complémentaire* d'un graphe G , noté $\bar{G}$, est tel que $V(\bar{G}) = V(G)$ et $E(\bar{G}) = \{\{u, v\} \mid e \notin E(G)\}$. Un *stable* est un sous-ensemble $S \subseteq V(G)$ de sommets de G tel que $G[S]$ est le complémentaire d'une clique, i.e le sous-graphe de G induit par S ne contient aucune arête. Un *couplage* est un sous-ensemble $E' \subseteq E(G)$ d'arêtes deux à deux disjointes de G . Un *couplage parfait* G' de G est un couplage tel que tous les sommets du graphe G sont incidents à exactement une arête de G' .

Chemins

Un *chemin* dans un graphe $G = (V, E)$ entre deux sommets $u \in V$ et $v \in V$ est une succession d'arêtes reliant u à v . On peut définir un chemin par un ensemble d'arêtes ou l'ensemble des sommets qu'il traverse, par exemple un chemin p reliant u à v peut être décrit par une liste d'arêtes $p = \{\{u, u_1\} \{u_1, u_2\}, \dots, \{u_n, v\}\}$ ou plus simplement par une liste de sommets $p = \{u, u_1, u_2, \dots, u_n, v\}$ tel que

$\{u, u_1\}, \{u_n, v\} \in E$ et $\forall i \in \{1, \dots, n-1\} \{u_i, u_{i+1}\} \in E$.

La longueur d'un chemin (lorsqu'elle n'est pas explicitement décrite) est le nombre d'arêtes du chemin, ou bien, dans le cas d'un graphe pondéré, la somme des poids des arêtes du chemin.

Voici quelques définitions courantes pour qualifier des caractéristiques spéciales de chemin. Un *chemin élémentaire* est un chemin ne passant pas deux fois par un même sommet. Un *chemin hamiltonien* est un chemin élémentaire qui passe par tous les sommets du graphe. Un *chemin simple* est un chemin ne passant pas deux fois par une même arête. Un *chemin eulérien* est un chemin simple qui passe par toutes les arêtes du graphe. Un *cycle* (aussi appelé boucle) est un chemin dont les deux extrémités sont identiques.

Classes de graphes

Certaines classes de graphes possèdent des propriétés structurelles ou algorithmiques intéressantes. Certains problèmes “difficiles” dans le cas général, peuvent être résolus efficacement sur certaines *classes de graphes* en utilisant leurs propriétés. Nous présentons ici quelques unes des principales classes de graphes liées à nos travaux de thèse.

Graphe biparti : Un graphe G est *biparti* si l'ensemble de ses sommets peut être partitionné en deux ensembles stables, i.e $V(G) = V_1 \cup V_2$ et $E(G) \subseteq \{\{u, v\} \mid u \in V_1, v \in V_2\}$.

Arbres et forêts : Un graphe ne contenant aucun cycle est une *forêt*. Si de plus ce graphe est connexe, le graphe est un *arbre*. Une forêt est donc composée de plusieurs composantes connexes, chacune d'entre elle étant un arbre. La classe des arbres a été largement étudié. On utilise quelque fois une notation spécifique pour les éléments d'un arbre qui diffère de celle des graphes quelconques. Ainsi, les sommets d'un arbre T sont appelés *nœuds* et les arêtes sont appelées *branches*. On distingue dans un arbre deux types de nœuds : les feuilles, qui correspondent aux sommets de degré 1, et les nœuds internes, qui correspondent aux autres nœuds de l'arbre. Un arbre est dit enraciné s'il existe une hiérarchie entre les nœuds, le nœud le plus haut dans la hiérarchie est appelé la *racine* de l'arbre. Chaque nœud de l'arbre dispose de zéro ou d'un nœud *père*, et de zéro, un ou plusieurs nœuds *fil*s. La racine est le seul nœud de l'arbre à n'avoir aucun père, et les feuilles sont les seuls nœuds à n'avoir aucun fils.

De nombreux problèmes difficiles pour les graphes quelconques, peuvent être résolus facilement sur les arbres et forêts. Néanmoins, il existe des problèmes qui restent difficiles même pour les arbres.

Graphe chemin : Le *graphe chemin* est un cas particulièrement simple d'un arbre ayant deux feuilles et dont les nœuds internes sont tous de degré 2.

Graphe parfait : Un graphe G est dit *parfait* si le nombre chromatique¹ et la taille de la plus grande clique de chaque sous-graphe induit sont égaux. Le théorème fort des graphes parfaits [34] s'énonce de la façon suivante :

Théorème 2.1.1. *Un graphe est parfait si et seulement si ni lui ni son complémentaire ne contiennent de cycle impair induit de longueur au moins cinq.*

De nombreux problèmes NP-Difficiles deviennent polynomiaux dans le cas de graphe parfait. En particulier, nous utilisons dans le chapitre 4, le fait que le problème du stable de poids maximum est polynomial pour cette classe de graphe [77].

Graphe d'intervalle : Un *graphe d'intervalle* est un graphe $G = (V, E)$ avec $V(G) = \{v_1, \dots, v_n\}$ tel qu'il existe un ensemble d'intervalles $\{I_1, \dots, I_n\}$ avec $E(G) = \{\{v_i, v_j\} \mid I_i \cap I_j \neq \emptyset\}$. Les graphes d'intervalles sont des graphes parfaits.

2.2 Complexité et algorithmique

En informatique fondamentale, l'efficacité d'un algorithme peut se mesurer par le nombre d'opérations qu'il réalise en fonction de la *taille de l'entrée*.

Le temps d'exécution d'un algorithme pour une entrée particulière est un nombre d'opérations élémentaires ou d'étapes à exécuter. Même si en pratique sur un ordinateur actuel le temps d'exécution de deux opérations basiques peut varier, nous le considérons constant d'un point de vue théorique.

Notation de complexité. Une notation asymptotique, popularisé par Knuth [108], est utilisée pour exprimer le temps d'exécution des algorithmes, ce qui permet de simplifier l'expression et met en évidence l'évolution d'un algorithme par rapport aux changements de la taille de l'entrée. Nous définissons ci-dessous la notation asymptotique utilisée dans ce mémoire. Soit $f(n)$ la fonction qui caractérise le

1. Le nombre chromatique d'un graphe est le nombre minimum de couleurs nécessaire pour colorier chaque sommet du graphe tel que tout sommets adjacents soient de couleurs différentes.

nombre maximum d'opérations pour l'exécution d'un algorithme sur une instance de taille n et soit $g(n)$ une fonction quelconque. Nous notons $f(n) = O(g(n))$ si $\exists k, n_0 \in \mathbb{R}^+$ tel que $\forall n \geq n_0, 0 \leq f(n) \leq kg(n)$. En d'autres termes, le temps d'exécution $f(n)$ de l'algorithme est bornée supérieurement par la fonction $g(n)$, à un facteur multiplicatif constant près. Lorsque la fonction $g(n)$ est un polynôme, l'algorithme est *polynomial*.

Problème d'optimisation/de décision. La théorie de la complexité est basée sur la notion de *problèmes de décision* pour lesquels la réponse est "OUI" ou "NON". Dans le cadre de cette thèse, nous considérons plutôt des *problèmes d'optimisation* qui consistent à minimiser ou maximiser une certaine fonction objectif à l'intérieur d'un espace de solutions. À chaque problème d'optimisation on peut faire correspondre un problème de décision qui consiste à déterminer s'il existe une solution pour laquelle la fonction objectif atteint une valeur donnée. De cette manière, une recherche dichotomique est suffisante pour trouver la solution optimale avec les réponses du problème de décision [38, 100]. Par exemple résoudre le problème de "trouver le chemin le plus court entre deux points sur une carte routière" est équivalent à résoudre que le problème de décision "existe-t-il un chemin de longueur inférieure à un nombre donné qui relie ces deux points de la carte routière".

Complexité classique. Nous allons présenter quelques notions importantes relatives à la complexité classique de calcul. Dans le cadre de notre travail de thèse, nous nous focalisons sur la complexité en temps d'exécution, mais le lecteur souhaitant approfondir la notion de complexité classique pourra consulter divers ouvrages de référence [9, 69, 131, 141].

Dès lors qu'un problème est décidable dans un certain modèle donné, il existe au moins un algorithme permettant de le résoudre : il s'agit de la recherche exhaustive d'une solution au problème. La notion de complexité d'un problème est liée à un modèle de calcul déterminé. Dans notre cas, il s'agit de la machine de TURING car, elle modélise le comportement d'un ordinateur qui exécute de manière séquentielle des instructions en passant d'un état à un autre. Une autre machine importante pour la complexité est la machine de TURING non-déterministe, qui peut être vue comme une machine de TURING pouvant se multiplier en passant d'un état à plusieurs autres états et continuer son exécution de manière parallèle. Des définitions plus formelles de ces machines sont disponibles dans les ouvrages cités précédemment.

Classes de complexité P et NP. La classe P et la classe NP sont les deux classes de complexité abordées dans ce mémoire. Ces deux classes sont en fait définies pour des problèmes de décision. La classe de complexité P (pour deterministic Polynomial) contient tous les problèmes pouvant être résolus par un algorithme polynomial sur une machine de TURING déterministe. La classe de complexité NP (pour Non-deterministic Polynomial) contient tous les problèmes pouvant être résolus en temps polynomial sur une machine de TURING non-déterministe. Il est trivial de montrer que $P \subseteq NP$ en n'utilisant pas l'aspect non-déterministe de la machine. Mais la question $NP \subseteq P$ et donc $P = NP$, reste ouverte et sa résolution serait une avancée majeure en informatique fondamentale, et aurait des conséquences sur de nombreux domaines scientifiques. De façon équivalente la classe NP peut être définie comme la classe des problèmes de décision dont les instances "OUI" peuvent être vérifiées en temps polynomial avec un certificat de longueur polynomial. Lorsque nous faisons référence en temps d'exécution, un temps polynomial indique un temps polynomial d'exécution par rapport à la taille de l'entrée sur une machine de TURING déterministe.

NP-complétude. La classe de complexité NP contient une sous-classe importante de problèmes : les problèmes NP-complets. Tout d'abord nous abordons la réduction polynomial qui permet d'établir une relation d'équivalence pour les problèmes de décisions, définie pour la première fois par KARP [100].

Definition 2.2.1. *Réduction de KARP (réduction polynomiale)*

Un problème Π_1 peut être réduit en temps polynomial (ou réduit polynomialement) à un problème Π_2 , s'il existe une fonction de réduction R d'une instance du problème Π_1 vers une instance du problème Π_2 , calculable en temps polynomial, et telle que l'instance I_1 du problème Π_1 a une réponse "OUI" si et seulement si l'instance I_2 du problème Π_2 a une réponse "OUI".

La fonction de réduction R effectue donc une transformation d'une instance du problème Π_1 vers une instance du problème Π_2 , en temps polynomial, tout en s'assurant que les réponses aux deux instances coïncident. La notion de réduction polynomiale permet donc de comparer la complexité des problèmes.

Nous pouvons maintenant définir les notions de problème NP-Difficile et NP-complet.

Definition 2.2.2.

Un problème Π_1 est dit NP-Difficile si et seulement si tout problème $\Pi_2 \in NP$, se réduit polynomialement à Π_1 .

Intuitivement, cela signifie que le problème Π_1 est au moins aussi difficile que n'importe quel problème de la classe NP .

Definition 2.2.3.

Un problème Π est dit *NP-Complet* si et seulement si, il est *NP-Difficile* et $\Pi \in NP$.

Autrement dit, si un problème est NP-Difficile et que l'on peut fournir un certificat polynomial pour la réponse "OUI", alors il est NP-Complet.

La première démonstration de NP-complétude [38] porte sur le problème de satisfiabilité couramment appelé SAT.

Rapidement, de nombreux autres problèmes ont été montrés NP-complets [69, 100]. Lorsque l'on souhaite montrer qu'un problème Π_1 est NP-difficile, il suffit de considérer un autre problème Π_2 connu comme étant NP-difficile, puis de montrer que Π_2 se réduit polynomialement à Π_1 (théorème de Cook [38]).

Problème: SAT	
Entrée:	Une formule booléenne $\psi(X)$ sur un ensemble de variables booléennes X
Question:	Existe-t-il une affectation des variables X telle que la formule $\psi(X)$ soit vraie ?

Algorithmes

Lorsque l'on s'intéresse à des problèmes d'optimisation, on peut distinguer trois principales familles d'algorithmes² en fonction de la qualité de la solution renvoyée.

Les *algorithmes exacts* garantissent l'optimalité des solutions renvoyées. Ces algorithmes sont utilisés sur les problèmes de la classe P . Il n'existe aucun algorithme polynomial exacte pour les problèmes NP-difficiles (à moins que $P = NP$). Ce type d'algorithme est donc peu utilisé pour leur résolution, même si les instances "réelles" du problème NP-difficile du voyageur de commerce sont résolues de nos jours dans un temps raisonnable.

Les *algorithmes d'approximation* ne garantissent pas l'optimalité mais borne la qualité de la solution par rapport à une solution optimale dans le pire des cas. On dit qu'un algorithme d'approximation a une *garantie de performance* $\rho(n)$ si, pour toute entrée de taille n , le coût C de la solution produite est éloignée de la solu-

2. Il existe aussi les algorithmes probabilistes qui donnent une valeur de l'espérance mathématique des solutions.

tion optimale C^* d'un facteur $\rho(n)$, c'est à dire $\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \geq \rho(n)$. On appelle *algorithme d'approximation facteur $\rho(n)$* un algorithme qui s'exécute en temps polynomial est atteint une garantie de performance de $\rho(n)$. Cette définition de garantie de performance s'applique aux problèmes de maximisation comme aux problèmes de minimisation. Pour le problème de maximisation, la solution C de l'algorithme sera inférieure ou égale à C^* la solution optimale et le facteur d'approximation sera de C^*/C . Pour le problème de minimisation, la solution C de l'algorithme sera supérieure ou égale à C^* la solution optimale et le facteur d'approximation sera de C/C^* . On considère que toutes les solutions sont positives, le facteur d'approximation est toujours supérieur ou égale à 1. Bien sûr un facteur de 1 donne une solution exacte. Les algorithmes d'approximation permettent de donner une solution aux problèmes NP-difficiles lorsque l'optimalité n'est pas requise.

Les *heuristiques* sont des algorithmes qui ne fournissent aucune garantie théorique. Elles sont utilisées quand le temps d'exécution des autres algorithmes est trop long ou qu'il n'existe pas d'algorithme offrant de garantie sur la qualité de la solution qu'il renvoie. L'évaluation de tels algorithmes est empirique et se base sur des tests expérimentaux. Généralement une heuristique est conçue pour un problème particulier et s'appuie sur sa structure propre. La stratégie appliquée fait abstraction des spécificités des problèmes on parle de *meta-heuristique*.

Dans cette thèse nous proposons deux heuristiques qui appartiennent chacune à une classe générique d'algorithmes : les algorithmes glouton et la recherche locale.

Les *algorithmes gloutons* construisent une solution de manière itérative, à chaque étape, un choix localement optimal est fait, il ne sera plus remis en cause jusqu'à la fin de l'algorithme. Pour certains problèmes spécifiques, il existe des algorithmes gloutons qui renvoient toujours une solution optimale.

Les *algorithmes de recherche locale* partent d'une solution initiale et la modifie de façon itérative en effectuant un "déplacement" vers une solution voisine. Cette méthode nécessite de choisir de façon appropriée les opérations élémentaires qui définissent la notion de voisinage. Le choix du déplacement repose uniquement sur des considérations locales et dépend seulement de la solution courante et des solutions voisines, d'où le terme de recherche locale. La différence avec un algorithme glouton réside dans le fait que les choix effectués ne sont pas définitifs. Une stratégie qui consiste à toujours choisir localement le meilleur voisin, peut conduire à un optimum local lorsque aucun voisin n'est meilleur que la solution courante. Pour s'extraire d'un optimum local, on s'autorise à détériorer la solution. Le risque est alors de

boucler en revenant sur l'optimum local précédent. Le mécanisme de mémoire qui consiste à interdire de revenir sur les dernières positions explorées nommée "Tabou" est destiné à éviter ces problèmes.

Programmation dynamique

Richard Bellman [21] est à l'origine du terme de *programmation dynamique* qui désigne les problèmes d'optimisation pouvant être résolus sous forme récursive.

Un algorithme de programmation dynamique se base sur une méthode de type *diviser pour régner*. Ceci consiste à décomposer le problème en sous-problèmes, chaque sous-problème pouvant être à son tour décomposer et ainsi de suite jusqu'à pouvoir être résolu de manière triviale. À la manière d'un arbre, un problème (père) peut être décomposé en sous-problèmes (fils), la racine de l'arbre étant le problème global, les sous-problèmes sont des nœuds internes et les sous-problèmes triviaux sont les feuilles de l'arbre. Pour finir on combine (recompose) les solutions des sous-problèmes fils pour obtenir la solution du problème père en partant des feuilles, et l'on remonte dans l'arbre jusqu'à parvenir à la racine et obtenir la solution globale.

Pour qu'un problème puisse être résolu par programmation dynamique il faut qu'il possède deux caractéristiques. La première est la présence de *sous-structures optimales* qui définissent les sous-problèmes dont les solutions optimales permettent de reconstituer une solution optimale pour le problème décomposé. En effet, la programmation dynamique construit la solution optimale à partir de solutions optimales de sous-problèmes. La seconde caractéristique est le *chevauchement des sous-problèmes*, c'est à dire qu'un même sous-problème apparaît plusieurs fois durant la décomposition du problème principal. Pour que l'utilisation de la programmation dynamique conduise à un algorithme polynomial, le nombre total de sous-problèmes distincts doit être polynomial par rapport à la taille de l'entrée.

Un *algorithme pseudo-polynomial* s'exécute en temps polynomial par rapport à la taille de l'entrée du problème lorsque les nombres sont représentés en base unaire. Intuitivement, l'algorithme s'exécute en temps polynomial si les nombres ne sont pas trop grands, par exemple s'ils sont bornées par une constante.

Quelques problèmes classiques de graphes

Nous présentons dans cette section quelques problèmes classiques sur les graphes auxquels nous faisons référence dans ce mémoire.

Problèmes de stable

Le *problème du stable maximum* consiste à trouver le plus grand ensemble de sommets deux à deux non adjacents dans un graphe G . Le problème de décision associé à ce problème d'optimisation est le suivant :

Problème: Stable Max

Entrée: Un graphe $G = (V, E)$, un entier $K \in \mathbb{N}$.

Question: Le graphe G contient-il un ensemble stable $S \subseteq V(G)$ de cardinalité au moins égale à K ?

Ce problème fait partie des vingt-et-un problèmes démontrés NP-complets dans [100].

Le *problème du stable de poids maximum* prend en entrée un graphe avec une pondération sur les sommets et consiste à trouver un stable dont la somme des poids des sommets est maximum.

Problème: Stable de poids maximum

Entrée: Un graphe $G = (V, E)$, un vecteur de poids $w \in \mathbb{N}^{|V|}$ et un entier $K \in \mathbb{N}$.

Question: Le graphe G contient-il un ensemble stable $S \subseteq V(G)$ dont la somme des poids des sommets est au moins égale à K ?

Comme pour le problème de stable maximum, le problème de stable de poids maximum est NP-complet dans le cas général. Si le graphe est parfait il existe un algorithme polynomial pour calculer un stable de poids maximum [77]. Nous utiliserons cette propriété dans le Chapitre 4.

Partition

Le *problème de partition* consiste à décider si un ensemble d'entiers positifs peut être partitionné en deux sous-ensembles A et B de telle sorte que la somme des nombres dans A est égale à la somme des nombres dans B .

Problème: Partition

Entrée: Un ensemble X de n entier $\{s_1, \dots, s_n\} \in \mathbb{N}^n$

Question: Existe-t-il une partition de X en deux sous-ensemble A et B , tels que

$$\sum_{x \in A} x = \sum_{x \in B} x ?$$

Le problème de partition est NP-complet [69]. Néanmoins, il existe un algorithme de programmation dynamique pseudo-polynomial pour le résoudre ainsi qu'une heuristique qui est efficace dans de nombreux cas concrets. Nous utiliserons ce problème pour des démonstrations de NP-difficulté dans le Chapitre 4.

2.3 Programmation Linéaire

Un *programmation linéaire* (PL) est un problème d'optimisation qui consiste à trouver parmi toutes les solutions d'un système d'inégalités linéaires une solution qui minimise ou maximise une fonction linéaire donnée. Une *solution réalisable* d'un programme linéaire est une affectation des variables qui satisfait toutes les contraintes du système d'inégalités linéaires. L'ensemble des solutions réalisables d'un PL est un *polyèdre convexe*, c'est à dire l'intersection d'un nombre fini de demi-espaces, chacun étant défini par une inégalité linéaire. La fonction linéaire à optimiser est appelée *fonction objectif* c'est une fonction affine à valeurs réelles définie sur ce polyèdre. Un algorithme de programmation linéaire doit donc trouver un point dans le polyèdre où cette fonction a la valeur la plus petite (ou la plus grande), si un tel point existe. Via une transformation standard les programmes linéaires peuvent toujours être exprimés sous la forme canonique suivante :

$$(P) \begin{cases} \text{Max} & c^T x \\ s.c & \\ & Ax \leq b \quad (1) \\ & x \geq 0 \quad (2) \end{cases}$$

Le vecteur $x \in \mathbb{R}^n$ représente les variables. L'objectif est de maximiser la fonction $c^T x$ définie par le vecteur de coefficients $c \in \mathbb{R}^n$. Les contraintes linéaires sont des inégalités de type inférieure ou égale. La matrice $A \in \mathbb{R}^{m \times n}$ précise les coefficients et le vecteur $b \in \mathbb{R}^m$ est appelé *le vecteur des seconds membres*. Les contraintes spéciales (2) sont appelées contraintes de non-négativité.

Les programmes linéaires peuvent être résolus en temps polynomial [104], et par conséquent appartiennent à la classe de complexité P , même si en pratique l'algorithme du *simplex* [42] est souvent utilisé alors qu'il n'est pas polynomial dans le pire des cas.

En raison de leur nature discrète, la modélisation de nombreux problèmes d'optimisation combinatoire nécessite que des variables de décision prennent des valeurs entières. On obtient alors un *programme linéaire en nombres entiers*. Si seul un sous-ensemble de variables sont contraintes à prendre des valeurs entières, on parle de *programmation mixte en nombres entiers*. Trouver une solution optimale d'un programme en nombres entiers (PLNE ou IP en anglais) ou d'un programmation linéaire mixte (MIP en anglais) est un problème NP-difficile.

Le problème d'optimisation d'un programme linéaire peut être formulé de la façon suivante :

Problème: Optimisation d'un programme linéaire	
Entrée:	Une fonction objectif $f : \mathbb{R}^m \rightarrow \mathbb{R}$, un système d'équation $Ax \leq b$ pour $x \geq 0$.
Question:	S'il existe, trouver un point x^* du polyèdre $P = \{x \in \mathbb{R}^n Ax \leq b, x \geq 0\}$ tel que $f = c^T x^*$ soit maximum ?

Dualité. À chaque programme linéaire sous forme standard est associé un programme linéaire particulier appelé son *dual*. Le dual du dual est le programme linéaire initial que l'on qualifie de *primal*. Un programme linéaire sous forme standard et son dual sont fortement reliés. La valeur de toute solution réalisable du dual est une borne supérieure de la valeur de la fonction objectif de toute solution primale (*dualité faible*). De plus, si le primal a une solution optimale, le dual a aussi une solution optimale et elles sont de mêmes valeurs (*dualité forte*). Le dual d'un programme linéaire sous forme standard s'écrit de la façon suivante :

$$(\bar{P}) \left\{ \begin{array}{ll} \text{Min} & b^T y \\ \text{s.c} & \\ & A^T y \geq c \\ & y \geq 0 \end{array} \right.$$

On peut voir que chaque contrainte primale est associé à une variable duale et réciproquement.

La dualité de la programmation linéaire est une notion très importante pour plusieurs raisons. Par exemple, elle permet d'obtenir un certificat d'optimalité pour une solution d'un programme linéaire. La dualité peut également être utile pour la conception ou l'analyse d'algorithmes, en particulier le cas d'algorithmes d'approximation [151].

Conditions des écarts complémentaires Les valeurs objectives d'une solution primale et d'une solution duale sont égales seulement si elles sont toutes les deux optimales. Les *conditions d'écarts complémentaires* sont des conditions nécessaires et suffisantes pour montrer qu'une solution réalisable du primal est optimale.

Théorem 2.3.1. *Une solution réalisable $x \in \mathbb{R}^n$ du problème primal est optimale si et seulement s'il existe une solution réalisable $y \in \mathbb{R}^m$ du problème dual telle que :*

$$\begin{cases} \forall i \in \{1, \dots, m\}, & y_i \left(b_i - \sum_{j=1}^n a_{ij} x_j \right) = 0, \\ \forall j \in \{1, \dots, n\}, & x_j \left(c_j - \sum_{i=1}^m y_i a_{ij} \right) = 0. \end{cases}$$

On en déduit deux conditions : (i) si une contrainte primale n'est pas serrée, la variable duale correspondante doit être nulle ; (ii) si une variable primale est positive alors, la contrainte duale associée est serrée.

Théorie Séparation/Optimisation. En s'appuyant sur l'algorithme de l'Ellipsoïde de Khachiyan [104], Grötschel, Lovász et Schrijver [78, 77] établissent une équivalence de complexité entre les problèmes de séparation et d'optimisation sur un polyèdre. Le problème de séparation sur un polyèdre s'énonce de la façon suivante :

Problème: Séparation sur un polyèdre

Entrée: Un polyèdre $P \subset \mathbb{R}^m$, un vecteur $y \in \mathbb{R}^m$.

Question: Le vecteur y appartient-il à P ? Sinon, donner un vecteur $c \in \mathbb{R}^m$ tel que $c^T y > c^T x$ pour tout $x \in P$.

Le problème de séparation consiste donc à déterminer si une solution y est valide pour le problème et si ce n'est pas le cas, à identifier une contrainte violée par y et satisfaite par toute solution $x \in P$.

Génération de colonnes

Considérons un programme linéaire sous forme standard. Si une solution optimale existe pour ce programme linéaire, la méthode simplex permet de calculer une solution optimale de base dans laquelle le nombre de variables non nulles, celles qui sont dans la base optimale, est au plus égal aux nombres de contraintes. Par conséquent, si le nombre de variables est très supérieur au nombre de contraintes, la grande majorité des variables, celles qui ne sont pas dans la base optimale sont à zéro.

La génération de colonnes consiste à itérer la méthode simplex sur un *programme linéaire restreint* qui est obtenu à partir du programme linéaire initial en sélectionnant un sous-ensemble de variables. A chaque itération de l'algorithme, on résout ce programme linéaire restreint et on résout un sous-problème, appelé *oracle de séparation*, qui consiste à tester s'il existe une variable x_j qui n'a pas été prise en compte et dont le coût réduit est positif (maximisation). De façon équivalente l'oracle de séparation revient à tester si la solution duale y associée à la base optimale du programme linéaire restreint vérifie toutes les contraintes du dual du programme linéaire initial. Les itérations cessent dès que l'oracle de séparation répond qu'il n'existe plus aucune variable de coût réduit positif.

L'optimalité de cet algorithme se démontre de la même façon que pour la méthode simplex, lorsque la condition d'arrêt est atteinte, la solution duale y associée à la base optimale est une solution réalisable duale du programme linéaire initial dont le coût coïncide avec celui de la solution primale associée à la base courante.

L'efficacité de cette technique repose sur le fait qu'en pratique un grand nombre de variables du programme linéaire initial ne sont jamais considérées. Par conséquent, la dimension du programme linéaire restreint est très réduite par rapport à celle du programme linéaire initial.

La génération de colonnes a été suggérée la première fois (en 1958) pour un problème de maximisation de flot avec demandes [64]. Elle fut ensuite utilisée pour les programmes linéaires avec structures décomposables [43] et le problème du découpage de stock [71]. Les applications actuelles de la génération de colonnes incluent les problèmes de tournées de véhicules [47, 48, 52], les problèmes d'ordonnancement modélisés par des graphes acycliques [97, 45, 122] et les problèmes de flots [16, 87].

Génération de colonnes pour le problème du Flot maximum. Nous allons maintenant présenter une utilisation de la méthode de génération de colonnes pour le problème de flot maximum. En principe, la résolution de ce problème, très étudié dans la

littérature, se fait via l'utilisation d'algorithmes combinatoires beaucoup plus performant. L'objectif est ici d'illustrer l'utilisation de la génération de colonnes sur un problème simple proche de celui que nous allons traiter dans le Chapitre 3.

Soit $G = (V, E)$ un graphe orienté avec pour chaque arc $e \in E$ une valeur non négative $C_e \geq 0$. Soit s et t deux sommets distincts appelés *source* et *puits*. Soit P l'ensemble des *st-chemins*, c'est à dire des chemins reliant s à t dans le graphe G . Pour tout chemin $p \in P$ la variable non négative ϕ_p représente la quantité de flot qui circule sur le chemin p . Il est dit réalisable s'il satisfait les contraintes suivantes :

$$\sum_{p \in P: e \in p} \phi_p \leq C_e, \quad \forall e \in E$$

Le problème de flot maximum peut être formulé de la façon suivante :

Problème: Flot maximum

Entrée: $G = (V, E)$ un graphe orienté, $C \in \mathbb{N}^{|E|}$, $s, t \in V$

Question: Quelle est la valeur maximale d'un flot réalisable entre de s à t ?

Nous allons décrire la méthode de génération de colonnes pour le problème de flot maximum. On note F le programme linéaire maître et $\bar{F}$ son dual. Soit $P' \subseteq P$ un sous-ensemble de chemins entre s et t . On considère le programme linéaire restreint à ce sous-ensemble $F(P')$:

$$F(P') \left\{ \begin{array}{ll} \text{Max} & \sum_{p \in P'} \phi_p \\ \text{s.c} & \\ & \sum_{\substack{p \in P' \\ e \in p}} \phi_p \leq C_e \quad \forall e \in E \\ & \phi_p \geq 0 \quad \forall p \in P' \end{array} \right.$$

On constate que chaque solution réalisable de $F(P')$ est réalisable pour $F(P)$ en mettant à zéro les variables des chemins n'appartenant pas à P' . Nous voulons donc savoir si la solution optimale de $F(P')$ l'est aussi pour $F(P)$, pour cela nous utilisons le dual restreint :

$$\bar{F}(P') \left\{ \begin{array}{ll} \text{Min} & \sum_{e \in E} \beta_e C_e \\ \text{s.c} & \\ & \sum_{e \in p} \beta_e \geq 1 \quad \forall p \in P' \\ & \beta_e \geq 0 \quad \forall e \in E \end{array} \right.$$

On peut déterminer si un sous-ensemble P' autorise une solution optimale pour P grâce à l'algorithme 1 qui vérifie que la solution β_e , $e \in E$ de $\bar{F}(P')$ est réalisable pour $\bar{F}(P)$. Pour cela on vérifie qu'aucun chemin $p \in P$ ne viole une contrainte en vérifiant que le plus court chemin dans le graphe à un poids supérieur à 1 pour la métrique β_e , $e \in E$. On remarque que le nombre de chemins portant du flot dans la solution optimale est borné par le nombre de contraintes $|E|$.

Algorithme 1: Résolution du problème de flot maximum par génération de colonnes

Données : $G = (V, E)$; $s, t \in V$; $C \in \mathbb{N}^{E(G)}$
Résultat : $(\phi_p : p \in \mathcal{P})$ une solution optimale du problème de flot maximum

- 1 $P' \leftarrow$ un ensemble arbitraire de st-chemins;
- 2 **répéter**
- 3 Calculer $(\beta_e : e \in E)$ une solution optimale $\bar{F}(P')$;
- 4 Calculer un plus court chemin $p \in \mathcal{P}$ avec la métrique β_e ;
- 5 **si** $\sum_{e \in p} \beta_e < 1$ **alors**
- 6 $P' \leftarrow P' \cup \{p\}$;
- 7 **fin**
- 8 **jusqu'à** ce qu'aucun chemin ne soit ajouté;
- 9 Renvoyer la solution optimale $(\phi_p : p \in \mathcal{P})$ de $F(P')$;

Programmation disjonctive

Nous définissons maintenant une méthode de modélisation qui nous permet dans le Chapitre 4 de modéliser le problème abordé sous la forme d'un programme linéaire, en se basant sur les travaux de Balas [14].

Un problème de *programmation disjonctive* (DP) est un problème de minimisation d'une fonction linéaire cx , $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$ et se modélise sous la forme normale disjonctive suivante :

$$DP = \begin{cases} \text{Min} & c^T x \\ \text{s.c} & \\ \bigvee_{i \in I} & \left(D_i x \geq d_i \right) \\ & x \geq 0 \end{cases}$$

Avec $d_i \in \mathbb{R}^m$, I un ensemble d'indice (pas nécessairement fini), la condition impose qu'il existe un indice $i \in I$ tel que x satisfasse au le système d'inégalités $D_i x \geq d_i$. Pour chaque $i \in I$ nous supposons que le polyèdre $\{x \in \mathbb{R}^n | D_i x \geq d_i\}$ est

non vide. Dans le cas contraire, nous pouvons omettre l'indice correspondant dans la formulation.

Le problème d'optimisation d'un programme disjonctif peut être formulé de la façon suivante :

Problème: Optimisation d'un programme disjonctif

Entrée: Une fonction objectif $f : \mathbb{R}^m \rightarrow \mathbb{R}$, un ensemble de systèmes d'in-égalités $D_i x \geq d_i$ pour $i \in I$.

Question: Si il existe, trouver un vecteur x^* qui satisfait au moins un système d'inégalités $D_i x \geq d_i$ pour $i \in I$ tel que $f = c^T x^*$ soit maximum ?

Programme linéaire équivalent à un programme disjonctif. Soit $Ens(F)$ l'ensemble des solutions réalisables de DP , c'est à dire l'ensemble des points qui satisfont les contraintes de DP . La forme normale disjonctive de $Ens(F)$ est :

$$Ens(F) = \left\{ x \in \mathbb{R}^n \mid \left(\bigvee_{i \in I} D_i x \geq d_i, x \geq 0 \right) \right\}$$

Soit $Poly(F_i)$ le polyèdre correspondant à l'indice $i \in I$:

$$Poly(F_i) = \left\{ x \in \mathbb{R}^n \mid (D_i x \geq d_i, x \geq 0) \right\}$$

L'ensemble $\mathcal{F}$ des solutions réalisables de DP est constitué de l'union de tous les polyèdres $Poly(F_i)$:

$$\mathcal{F} = \bigcup_{i \in I} Poly(F_i)$$

Nous définissons l'enveloppe convexe de $\mathcal{F}$ à condition que I soit fini et que le cône de récession de chaque $Poly(F_i)$ soit identique. Si tous les $Poly(F_i)$ sont bornés alors leur cône de récession est $\{0\}$.

$$clconv \mathcal{F} = \left\{ x \in \mathbb{R}^n \mid \begin{array}{l} x = \sum_{i \in I} x^i \\ D^i x^i \geq d^i \lambda^i, \quad \forall i \in I \\ \sum_{i \in I} \lambda^i = 1, \\ x^i \geq 0, \quad \forall i \in I \\ \lambda^i \geq 0, \quad \forall i \in I \end{array} \right\}$$

Le résultat fondamental qui permet de formuler un programme disjonctif sous la forme d'un programme linéaire est le suivant :

Corollary 2.3.1. *Si le programme linéaire F suivant a un optimum fini, alors le programme disjonctif (DP) est équivalent à F .*

$$F = \left\{ \begin{array}{l} \text{Min} \quad \sum_{i \in I'} x^i \\ \text{s.c} \\ D^i x^i \geq d^i \lambda^i, \quad \forall i \in I' \\ \\ \sum_{i \in I} \lambda^i = 1, \\ \\ x^i \geq 0, \quad \forall i \in I' \\ \lambda^i \geq 0, \quad \forall i \in I' \end{array} \right.$$

Dans le sens où :

- (i) Si x est un point de $\text{clconv}F$, alors il existe $k \in I$ tel que χ , définit par $(x^k, \lambda^k) = (x, 1)$ et $(x^h, \lambda^h) = (0, 0), \forall h \in I - \{k\}$, est un point appartenant à P l'ensemble réalisable de F .
- (ii) Si χ est un point de P , alors il existe $k \in I$ tel que $\lambda^k = 1, (x^h, \lambda^h) = (0, 0), \forall h \in I - \{k\}$, et $x = x^k$ est un point de $\text{poly}(F_k)$
- (iii) x est une solution optimale de DP si et seulement si $x^k = x$ et $\lambda^k = 1$ est une solution optimale de F ($k \in I$ défini dans (i)).

Démonstration 2.3.1.

La preuve de ce corollaire se base sur le fait que toute solution optimale de DP est un point extrême de $\text{clconv}\mathcal{F}$. Soit x^* une solution optimale de DP . Il existe $i^* \in I$ tel que $D_{i^*} x^* \geq d_{i^*}$ (par définition). Si x^* point extrême de $\text{Poly}(F_{i^*})$ mais pas de $\text{clconv}\mathcal{F}$. Alors il existe y point extrême de $\text{clconv}\mathcal{F}$ tel que $c^t y < c^t x^*$. Or y est une solution de DP car les sommets de $\text{clconv}\mathcal{F}$ sont des sommets des polyèdres $\text{Poly}(F_i)$. Donc y est meilleur que x^* ce qui contredit l'hypothèse initiale.

□

Notes - État de l'art de la programmation disjonctive. Plusieurs cas particuliers de programmation disjonctive ont été étudiés dans le passé [13, 73, 130]. La formulation en programmation disjonctive est utile pour les structures de problèmes provenant de conditions physiques “logiques”. La programmation disjonctive a surtout été étudiée pour les programmes linéaires en nombres entiers [12, 14, 76, 139].

2.4 Problème de flots et variantes

Le problème d'acheminement des marchandises dans un réseau plus connu comme le problème de flot maximum, a été étudié à la fin des années 1950 généralisent le théorème de Menger [121] qui établit une égalité entre la cardinalité minimum d'une coupe séparant deux sommets d'un graphe et le nombre maximum de chemins disjoints entre ces sommets. De manière simultanée, [63] et [57] généralisent ce théorème et définissent un algorithme pour obtenir la valeur optimale du problème du flot maximum et de la coupe minimum.

La formulation en programme linéaire utilisant des chemins est très intuitive, mais elle nécessite l'utilisation d'une variable pour chaque chemin ce qui est, en générale, non polynomial par rapport à la taille du graphe. Une autre formulation consiste à introduire une variable pour représenter la quantité de flot qui circule sur chacun des arcs ou des arêtes du graphe. On impose alors une *contrainte de conservation* du flot en chaque sommet du graphe à l'exception de la source et du puits. La contrainte de conservation impose que la somme du flot sur les arcs entrants dans un sommet doit être égale à la somme du flot sur les arcs sortant.

Il existe donc deux types de formulation pour les problème de flot. Les formulations *arcs-chemins* utilisant des variables représentant la quantité de flot qui circule sur un chemin. Les formulations *sommets-arcs* utilisant des variables qui représentent la quantité de flot qui circule sur les arcs du réseau. Les formulations sommets-arcs comportent un nombre de variables et de contraintes polynomial par rapport à la taille du graphe en entrée et permet d'obtenir une solution optimale en temps polynomial. De plus si les capacités des arcs sont entières, la matrice des contraintes étant totalement unimodulaire, cette propriété permet d'obtenir naturellement une solution optimale entière par résolution du programme linéaire.

Problèmes de multiflot

La notion de flot permet de modéliser les situations où un seul produit circule dans un réseau. Lorsque plusieurs produits circulent en se partageant les capacités des liaisons, on doit faire appel à la notion de *multiflot*. On associe à chaque produit une *demande*, qui est le nombre d'unités devant être acheminées à travers le réseau entre une paire source-destination. On caractérise une demande $k \in \mathcal{K}$ pour le triplet $k = (s_k, t_k, d_k)$ avec $s_k, t_k \in V$ et $d_k \in \mathbb{N}$ la valeur du flot demandée. La valeur d'un flot représente la quantité de flot qui sort du sommet source ou la quantité qui entre dans le puits. Le problème de multiflot peut être formulé de la façon suivante :

Problème: Multiflot

Entrée: $G = (V, E)$ un graphe orienté, $C \in \mathbb{N}^{|E|}$, des triplets (s_k, t_k, d_k) avec $s_k, t_k \in V$ et $d_k \in \mathbb{N}$ pour chaque demande $k \in \mathcal{K}$

Question: Si il existe, trouver un flot pour chaque demande $k \in \mathcal{K}$ entre les sommets s_k et t_k de valeur d_k (tel que la somme des valeurs des flots circulant sur chaque arc $e \in E$ soit inférieure ou égale à sa capacité C_e) ?

Notes - État de l'art du problème de multiflot. Dans les problèmes de multiflot linéaires de coût minimum, un coût est attribué à chaque lien et la fonction objectif consiste à minimiser la somme des coûts de routage [65]. Le problème interdisant de répartir le flot entre différents chemins a également été étudié [20]. Le problème de multiflot a été largement étudié et a donné naissance à un large éventail de variantes dues aux différents contextes dans lesquels une modélisation utilisant la notion de multiflot est possible. Beaucoup de problèmes de distribution et d'ordonancement peuvent être exprimés et résolus comme des problèmes de multiflot. La plupart des problèmes de routage dans les réseaux de télécommunications peuvent naturellement être modélisés comme un problème de multiflot. Les problèmes de télécommunications ont également amené à considérer des coûts non-linéaires [26, 66]. Par exemple, l'un des problèmes importants dans la conception des réseaux informatiques à commutation de paquets consiste à déterminer des itinéraires sur lesquels les paquets doivent être transmis pour obtenir une qualité de service optimale. Dans la littérature le délai moyen d'acheminement des messages est le critère le plus fréquemment utilisé pour mesurer cette qualité de service. Pour finir, il existe une importante bibliographie sur la relation d'équivalence entre la multi-coupe et le multi-flot [24, 40].

Problème de flot généralisé

La notion de *flot généralisé* est une généralisation de la notion de flot maximal dans laquelle chaque arc $e \in E$ du réseau $G = (V, E)$ a un facteur de gain α_e . Si x unités de flot entre dans l'arc à e , alors $x\alpha_e$ unités de flot en sorte. On peut formuler le problème de flot généralisé de la façon suivante :

Problème: Flot généralisé

Entrée: Un graphe orienté $G = (V, E)$, $C \in \mathbb{N}^{|E|}$, $\alpha \in \mathbb{R}_+^{|E|}$, $s, t \in V$.

Question: Quelle est la valeur maximale du flot réalisable allant de s à t en prenant en compte le gain subi par le flot lors du passage dans chacun des arcs de G ?

Un réseau pour lequel les gains des arcs sont inférieurs ou égaux à 1 est appelé *lossy*. Dans le cas de réseau *lossy*, on ne peut pas avoir de cycle générateur de flot, c'est à dire de cycle $\mathcal{C}$ tel que $\prod_{e \in E(\mathcal{C})} \alpha_e > 1$. L'existence de tel cycle complique la résolution des problèmes de flot généralisé car il faut alors considérer chaque sommet de ce cycle comme une source potentielle de flot.

La formulation arc-chemin des problèmes de flot généralisé fait intervenir une variable ϕ_p pour chaque chemin p qui représente la quantité de flot émise par la source qui circule sur le chemin p .

$$F \left\{ \begin{array}{ll} \text{Max} & \sum_{p \in P} \phi_p \\ \text{s.c} & \\ & \sum_{p \in P} x_p(e) \phi_p \leq C_e \quad \forall e \in E \\ & \phi_p \geq 0 \quad \forall p \in P \end{array} \right.$$

P l'ensemble des st -chemins du graphe, c'est-à-dire l'ensemble des chemins reliant s à t dans G . Pour chaque arc $e_q \in p := (e_1, e_2, \dots, e_r)$, on note $x_p(e_q) := \prod_{i=1}^q \alpha_{e_i}$ la quantité de flot qui sort de l'arc e_q lorsqu'une unité de flot est émise à partir de s et transite sur les arcs $e_1, e_2, \dots, e_{q-1}$.

On remarque que si l'on veut appliquer la méthode de génération de colonnes pour résoudre ce programme linéaire, l'oracle de séparation revient à résoudre un problème de plus court chemin avec une fonction de longueur spécifique. En effet, cette longueur de chemin n'est plus la simple addition des longueurs des arcs du chemin, mais une composition successive de fonctions croissantes.

Notes - État de l'art du problème de flot généralisé. Le problème de flot généralisé a été largement étudié dans les années 1960 et au début des années 1970 [73, 124, 125, 145]. Ce problème peut être utilisé pour modéliser de nombreuses situations qui sont impossibles à modéliser avec la notion de flot "standard" [72, 114]. Le

facteur de gain peut être utilisé pour représenter le fait qu’une certaine fraction de la marchandise est perdue pendant le transport ou que la marchandise qui entre dans un arc se transforme en une autre marchandise différente avant de le quitter, pour modéliser les procédés de fabrication, de change, etc. Le problème de flot généralisé est à bien des égards similaires au problème de flot de coût minimum, où chaque arc a un coût par unité de flot en plus de sa capacité [125, 145]. Les deux problèmes peuvent être vus comme un problème de transport de marchandise d’un producteur à un consommateur. Le problème de flot généralisé peut être résolu par programmation linéaire [56] et par des algorithmes combinatoires [74] qui se basent sur l’idée de l’annulation des cycles négatifs [75, 105].

Problème de flots sur les chemins de longueur bornée

Ces problèmes prennent en entrée un graphe dont les arêtes sont munies de longueurs. Le flot ne peut circuler que sur des chemins dont la longueur est bornée. On obtient la formulation suivante :

Problème: Flot sur les chemins de longueur bornée	
Entrée:	Un graphe orienté $G = (V, E)$, $C \in \mathbb{N}^{ E }$, une longueur d’arc $l \in \mathbb{N}^{ E }$, $s, t \in V$, $B \in \mathbb{N}$ une longueur maximale.
Question:	Quelle est la valeur maximum d’un flot réalisable allant entre s et t tel que la longueur des chemins supportant le flot soit inférieure ou égale à B ?

Pour pouvoir formuler ce problème comme un programme linéaire, on peut utiliser une formulation arc-chemin :

$$PL \left\{ \begin{array}{ll} \text{Max} & \sum_{p \in P(B)} \phi_p \\ s.c & \\ & \sum_{p \in P(B)} \phi_p \leq C_e \quad \forall e \in E \\ & \phi_p \geq 0 \quad \forall p \in P(B) \end{array} \right.$$

où $P(B)$ est l’ensemble des st -chemins dont la longueur est inférieure ou égale à B et ϕ_p la valeur du flot sur le chemin $p \in P(B)$.

On remarque que si l’on veut appliquer la méthode de génération de colonnes pour résoudre ce programme linéaire, l’oracle de séparation est un problème NP-

difficile de plus court chemin avec plusieurs longueurs où l'on cherche à minimiser une longueur tandis que la seconde ne doit pas excéder la borne B (problème [ND30] de [69]).

Notes - État de l'art du problème de flot sur les chemins de longueur bornée. À notre connaissance, les premiers travaux effectués sur les problèmes de flots sur les chemins de longueur bornée apparaissent comme généralisation du problème de flot maximum [2] ou comme sous-problème particulier de flot entier [118, 60]. Des algorithmes efficaces ont été développés pour des longueurs de chemins inférieures ou égales à 3 [94, 120]. Ce problème a été montré NP-difficile même dans les graphes planaires [27, 120, 147]. Un schéma d'approximation entièrement polynomiale est décrit pour le problème de flot fractionnaire sur les chemins à longueur bornée [10]. Le problème de flot fractionnaire sur les chemins à longueur bornée unitaire peut être résolu en temps polynomial en utilisant la programmation linéaire [10, 109]. Ces problèmes de flot et de chemin de longueur bornée apparaissent naturellement dans de nombreux problèmes réels d'optimisation ayant un grand nombre de données, c'est pourquoi des heuristiques ont aussi été développées [136, 29, 149, 89].

2.5 Problème de plus court chemins et variantes

Le problème de plus court chemin est bien connu depuis les années 1950 [22, 62]. L'algorithme de Dijkstra [55] qui le résout est devenu une référence dans le domaine de la recherche opérationnelle et plus généralement en informatique. Le problème de plus court chemin peut être formulé de la façon suivante :

Problème: Plus court chemin

Entrée: Un graphe orienté $G = (V, E)$, une longueur l_e pour chaque arc $e \in E$, deux sommets $s, t \in V$.

Question: Trouver un chemin de longueur minimale reliant s à t

Dans cette section nous présentons des algorithmes pour résoudre des variantes du problème de plus court chemin. Tous peuvent être vus comme des algorithmes de programmation dynamique et sont similaires à l'algorithme de Dijkstra.

Plus court chemin généralisé

Dans les réseaux de transport d'énergie ou dans les transactions financières il n'est pas rare que le flux transporté subisse des gains ou des affaiblissements. Pour rappel, les réseaux généralisés (avec gain) se modélisent par un graphe $G = (V, E)$, chaque arc $e \in E$ a un facteur de gain α_e en plus de sa capacité C_e . Le variation du flot dû au gain est prise en compte de la façon suivante : si x unités de flot entre dans l'arc e , alors $x\alpha_e$ unités de flot en sorte. Le coût de l'envoi d'une unité de flot jusqu'à t sur le st-chemin $p \in \mathcal{P}$ est appelé la longueur généralisée du chemin p .

Le problème de plus court chemin généralisé peut être formulé de la façon suivante :

Problème: Plus court chemin généralisé	
Entrée:	Un graphe orienté $G = (V, E)$, $C \in \mathbb{N}^{ E }$, un coût par arête $l \in \mathbb{N}^{ E }$, $\alpha \in \mathbb{R}_+^{ E }$, $s, t \in V$, .
Question:	S'il existe, trouver un chemin reliant s à t de coût généralisé minimum.

On remarque que si tous les gains sont égaux à un alors le coût généralisé du chemin est égal au coût du chemin et ce problème se réduit à trouver le plus court chemin "traditionnel".

Comme pour le problème de flot généralisé nous supposons que le graphe est lossy, c'est à dire que tous les gains sont inférieurs ou égaux à 1. En imposant cette condition on évite le problème suivant. Soit p un chemin qui supporte ϕ unités de flot émises par s et achemine une unité de flot à t . Si p a une boucle génératrice de flot, on peut remplacer p par le chemin p^* obtenue à partir de p en effectuant une boucle de plus. Cette boucle étant génératrice de flot, pour une même unité de flot acheminé jusqu'à t le flot émis par la source sur p^* sera inférieur au flot émis par la source sur p . Par conséquent le chemin p^* qui effectue une boucle de plus est plus économique que p . Par conséquent nous pouvons remplacer p par p^* pour obtenir une meilleur solution. Nous pourrions répéter ainsi l'opération indéfiniment jusqu'à obtenir un flot asymptotiquement nul émis par s . L'existence de cycles générateurs de flots pour le problème de plus court chemin généralisé est analogue au problème que représente l'existence de cycles de coûts négatifs pour le problème de plus court chemin traditionnel.

Une formulation classique du problème de plus court chemin généralisé, sous la

forme d'un programme linéaire, consiste à le formuler comme un problème de flot généralisé de coût minimum acheminant une unité de flot à t .

$$PCCG = \begin{cases} \text{Min} & \sum_{e \in E} l_e \times \phi_e \\ \text{s.c} & \\ & \sum_{\{u,v\} \in E} \alpha_{\{u,v\}} \phi_{\{u,v\}} - \sum_{\{v,w\} \in E} \phi_{\{v,w\}} = \begin{cases} 1 & \text{if } v = t \\ 0 & \text{if } v \in V \setminus \{s, t\} \end{cases} \\ & \phi_{(u,v)} \geq 0, \forall (u,v) \in E \end{cases}$$

L'algorithme utilisé dans [61] pour résoudre ce problème est un algorithme de programmation dynamique similaire à l'algorithme de Dijkstra. Cet algorithme construit le plus court chemin généralisé en *étendant* successivement des chemins.

Definition 2.5.1. Soit un graphe $G = (V, E)$, $s, v \in V$ et p un chemin entre s et v . On étend un chemin p par un arc $e = \{v, w\} \in E$ en construisant le chemin p' égal au chemin p auquel on ajoute l'arc e .

Nous décrivons maintenant la fonction de longueur pour un arc dans un graphe généralisé. Pour rappel le plus court chemin généralisé doit faire parvenir une unité de flot à t de la manière la plus économique. La fonction de longueur des arcs se base sur l'observation suivante. Si faire arriver une unité de flot sur un chemin reliant s à u coûte x alors on peut faire arriver α_{uv} unité de flot en v avec un coût $x + l_{\{u,v\}}$ via l'arc $\{u, v\} \in E$. Or nous souhaitons faire parvenir une unité de flot en v , en mettant à l'échelle, on obtient qu'une unité de flot en v coûte $\frac{x + l_{\{u,v\}}}{\alpha_{\{u,v\}}}$. Donc, si x est la longueur généralisée d'un chemin jusqu'à u , la fonction de x qui permet de calculer la longueur généralisée d'un chemin jusqu'à v est $x \mapsto \frac{x + l_{\{u,v\}}}{\alpha_{\{u,v\}}}$.

L'optimalité de l'algorithme de programmation dynamique résolvant le problème de plus court chemin généralisé se base sur la proposition suivante :

Proposition 2.5.1. *Si l'arête $\{u, v\}$ est la dernière arête d'un plus court chemin généralisé p entre s et v alors p est constitué d'un plus court chemin généralisé entre s et u suivie de l'arête $\{u, v\}$*

La preuve de ce résultat est immédiate en utilisant le fait que la fonction qui définit la longueur du chemin jusqu'à v en fonction de la longueur du chemin jusqu'à u est une fonction croissante.

Cette proposition montre que seuls les plus courts chemins généralisés entre s et un autre sommet du graphe sont à considérer pour obtenir une solution optimale.

Dans l'algorithme que nous décrivons nous introduisons la notion de *label*. Un label caractérise un chemin entre s et un sommet du graphe. Un label fait *référence* à un sommet $v \in V$ si il représente un chemin entre s et v . Un label est constitué du sommet auquel il fait référence et de la longueur du chemin qu'il représente. L'opération d'extension d'un label dont le sommet de référence est u par l'arête $\{u, v\}$ consiste à créer un nouveau label correspondant au chemin représenté par L étendu par l'arc $\{u, v\} \in E$. Par abus de langage on utilise le mot label pour désigner le chemin qu'il représente quand on parle de la longueur d'un label.

Definition 2.5.2. *Soit L_1 un label de longueur x_1 et L_2 un label de longueur x_2 , on note $L_1 \leq L_2$ la relation L_1 domine L_2 qui est vrai si et seulement si L_1 et L_2 ont le même sommet de référence et $x_1 \leq x_2$.*

La forme de l'algorithme ci dessous est légèrement différente de celle couramment présentée. Nous utilisons cette forme pour montrer la similitude avec l'algorithme 3, présenté dans la section suivante. Dans le chapitre 3, nous utilisons un algorithme qui généralise dans un certain sens ces deux algorithmes.

Algorithme 2: Plus court chemin généralisé de s à t

Données : $G = (V, E)$; $s, t \in V(G)$; α_e et l_e pour tout $e \in E$
Résultat : Un plus court chemin généralisé de s à t

```

1 Soit  $L_s$  le label du chemin  $p = \{s\}$  de longueur nulle ;
2  $\mathcal{L} = L_s$  ;
3  $Q.put(L_s)$  ;
4 tant que  $Q \neq \emptyset$  faire
5   Soit  $L_{min} \leftarrow Q.pop()$  ;
6   Soit  $u$  le sommet de référence de  $L_{min}$  ;
7   pour tous les voisins  $v$  de  $u$  faire
8     Soit  $L'$  le label  $L_{min}$  étendu par l'arc  $\{u, v\}$  ;
9     si  $\nexists L^* \in \mathcal{L}$  tel que  $L^* < L'$  alors
10        $Q.put(L')$  ;
11        $\mathcal{L} = L' \cup \mathcal{L}$  ;
12     fin
13     si  $\exists L^* \in Q$  tel que  $L' < L^*$  alors
14        $Q.remove(L^*)$  ;
15     fin
16   fin
17 fin
18 Renvoyer le chemin associé au label en  $t$ .
```

L'algorithme 2 est un algorithme de type *label-correcting* qui permet de résoudre le problème de plus court chemin généralisé. À chaque étape de cet algorithme nous disposons de $\mathcal{L}$ l'ensemble de tous les labels déjà créés et d'un tas Q contenant des labels susceptibles d'engendrer la solution optimale. On extrait de Q le label L_{min} de coût minimum et on l'étend à tous les voisins de son sommet de référence. Pour chaque label L' ainsi généré, on vérifie qu'il n'existe pas un label de $L^* \in \mathcal{L}$ tel que $L^* \leq L'$. Si tel n'est pas le cas on ajoute L' à $\mathcal{L}$ et à Q . L'algorithme se termine quand le tas Q est vide.

La preuve de cet algorithme est similaire à celle de l'algorithme de Dijkstra et consiste à montrer, grâce à la proposition 2.5.1, que à chaque itération de l'algorithme le label L_{min} de coût minimum retiré du tas est un plus court chemin généralisé entre s et le sommet de référence de L_{min} .

Nous abordons maintenant la complexité de notre algorithme. Tout d'abord nous analysons la structure de données utilisée. Cette structure est un tas, noté Q , sur

lequel on effectue deux opérations. L'opération *push* ajoute un élément et l'opération *pop* renvoie le plus petit élément du tas et le supprime. Ces deux opérations se font en un temps logarithmique par rapport à la taille du tas. Dans notre algorithme la taille maximale du tas est $|V|$, i.e un label pour chaque sommet du graphe. Donc la complexité pour toutes opérations sur le tas dans notre algorithme est en $O(\log|V|)$.

À chaque étape de l'algorithme on considère le label de longueur minimum, ce label a un sommet de référence. Nous affirmons que plus aucun label ayant ce sommet de référence n'apparaîtra dans le tas. En effet, le label de longueur minimum correspond au chemin le plus court entre s et ce sommet de référence. Pour qu'un autre label apparaisse dans le tas il faudrait qu'il corresponde à un chemin encore plus court, ce qui est impossible. Par conséquent chaque sommet du graphe n'est considéré qu'une seule fois. Pour chaque sommet, le nombre d'opérations effectuées est au plus égal au degré sortant du sommet (extension du label). Si l'on somme sur tous les sommets cela donne $O(|E| + |V|)$ opérations d'extension de label. Pour finir, les opérations de comparaisons de labels se font en temps constant. La complexité de l'algorithme est donc en :

$$O((|E| + |V|) \times \log|V|)$$

Notes - État de l'art du plus court chemin généralisé Les réseaux généralisés ont déjà été étudié. En particulier, le problème de flot généralisé de coût minimum est décrit dans [3]. La méthode couramment utilisée pour résoudre le problème de plus court chemin généralisé [5, 35, 36, 86] consiste à le formuler comme un programme linéaire, à résoudre son problème dual et à construire une solution du primal en fonction de la solution duale. Le programme linéaire dual a la particularité d'avoir seulement deux variables par inégalité ce qui simplifie sa résolution.

Plus court chemin multi-contraints

Nous présentons dans cette section un problème que nous rencontrons lors de la résolution d'un problème industriel dans le chapitre 3. Ce problème apparaît dans la résolution par génération de colonnes d'un problème de flot généralisé où les chemins qui portent du flot doivent satisfaire plusieurs contraintes de longueurs sur les chemins.

Le problème de plus court chemin multi-contraint est un cas particulier du problème de plus court chemin avec contraintes de ressources. La méthode de résolution que nous proposons est valide pour les deux problèmes. D'ailleurs nous utilisons le

terme de ressources qui est représentatif du problème général au lieu de pondération qui est plus naturel pour le problème restreint.

Soit un graphe orienté $G = (V, E)$. L'ensemble des sommets V comprend une source s et une destination t , qui correspondent respectivement au sommet de départ et d'arrivée du chemin. On définit un ensemble de ressources $\mathcal{R}$. Une *ressource* $r \in \mathcal{R}$ représente une unité de mesure, comme le temps ou un coût. À chaque paire constituée d'une arête $e \in E$ et d'une ressource $r \in \mathcal{R}$, on associe $r(e)$ une fonction de consommation de ressource appelée *REF*. Une *REF* représente la quantité de ressource consommée pour transiter sur l'arc e . Ici nous supposons que les *REF* sont des fonctions entières croissantes. Pour chaque chemin $p = \{e_1, \dots, e_n\}$ on associe un vecteur de ressources qui correspond aux ressources consommées par p pour traverser les arcs $\{e_1, \dots, e_n\}$. Nous définissons pour chaque ressource $r \in \mathcal{R}$ une borne b_r pour la consommation de la ressource r . Un chemin p est *valide* si et seulement si pour chaque ressource r sa consommation est inférieure ou égale à b_r . Parmi les ressources nous définissons $r^* \in \mathcal{R}$ la ressource à minimiser. La fonction *REF* correspondant à la ressource r^* peut être réelle. Pour chaque ressource $r \in \mathcal{R}$ on note W_r la taille de l'ensemble pour lequel la ressource est définie. Si la fonction de ressource est entière alors $W_r = b_r$. On suppose que l'ensemble des ressources $\mathcal{R}$ est ordonné en fonction des W_r pour tout $r \in \mathcal{R}$, c'est à dire que $W_i \leq W_{i+1}$. On

note $W = \prod_{i=1}^{|\mathcal{R}|-1} W_i$.

Le *problème de plus court chemin multi-contraint* peut être formulé de la façon suivante :

Problème: Plus court chemin multi-contraint

Entrée: Un graphe $G = (V, E)$, un ensemble de ressources $\mathcal{R}$, $r^* \in \mathcal{R}$ la ressource à minimiser, pour chaque ressource $r \in \mathcal{R}$ et pour chaque arête $e \in E$ une *REF* croissante, une borne b_r et deux sommets $s, t \in V$

Question: Si il existe, trouver un chemin valide reliant s à t qui minimise r^* .

Le problème du plus court chemin multi-contraint est NP-Complet, même dans le cas d'une seule contrainte (problème [ND30] de [69]).

Definition 2.5.3. Soit deux vecteurs de réel $x = (x_1, \dots, x_n)$, $y = (y_1, \dots, y_n)$. On note $x \leq y$ la relation y est dominé par x (ou x domine y) si et seulement si $\forall i \in [1, \dots, n] \ x_i \leq y_i$.

L’algorithme de label-correcting L’algorithme pseudo-polynomial que nous proposons pour résoudre le problème de plus court chemin multi-contraint est basé sur la programmation dynamique au même titre que l’algorithme de label-correcting classique décrit dans [49] qui résout ce problème lorsque les multiplicateurs sont tous égaux à un. Cet algorithme est souvent considéré comme le plus efficace [96]. Notre algorithme maintient un ensemble de chemins non-dominés (en construction) susceptibles d’engendrer une solution optimale.

Comme pour l’algorithme 2 précédent, nous utilisons des labels. Un label est constitué d’un sommet de référence et d’un vecteur de ressources consommées par le chemin qu’il représente. Comme précédemment la notion d’extension du label reste inchangée. La notion de domination se définit sur les labels en prenant leur vecteur de ressource comme vecteur de comparaison. Bien entendu seul des labels ayant le même sommet de référence peuvent être comparés.

Notre algorithme réduit le nombre de labels créés lors de son exécution évitant d’étendre des labels qui sont dominés. La domination permet de réduire le nombre de labels étendus tout en préservant suffisamment de label pour obtenir une solution optimale. Sans la suppression des labels dominés l’algorithme calculerait tous les chemins possibles entre s et t . L’algorithme que nous proposons se base sur la proposition suivante.

Proposition 2.5.2. Soient L_1 et L_2 deux labels représentant les chemins P_1 et P_2 avec $L_1 \leq L_2$. Soit L le label d’un chemin P commençant par P_1 alors le label L' du chemin obtenu à partir P en remplaçant P_1 par P_2 domine L .

Démonstration 2.5.1.

Les chemins P et P' sont identiques à partir du sommet de référence u commun à L_1 et L_2 . En u , le vecteur de ressources de L (celui de L_1) est dominé par le vecteur de ressources de L' (celui de L_2). La monotonie des fonctions de consommation de ressources assure que cette propriété est préservée lorsque l’on parcourt la fin du chemin P commun aux labels L et L' .

□

Nous avons donc un critère pour exclure certains labels de l'exécution de l'algorithme car lorsqu'un label est dominé, son retrait ne supprime pas de solution optimale.

Remarque L'algorithme que nous proposons marche pour toute *REF* entière respectant cette condition, même si la *REF* prend plusieurs ressources en paramètre. Elle doit seulement être croissante si on fixe les autres ressources.

Définition 2.5.4. *Soit un label L représentant un chemin p atteignant le sommet u . Un sommet v voisin de u atteignable pour le label L si et seulement si le chemin p' composé du chemin p et de l'arête $\{u, v\}$ est valide.*

La complexité de l'algorithme se base sur l'hypothèse suivante.

Hypothèse 2.5.1. *Soit $\mathcal{L}$ ensemble de label, nous supposons qu'il est possible de choisir un label $L_{min} \in \mathcal{L}$ tel qu'aucun label obtenu par extension d'un label $L \in \mathcal{L}$ ne domine L_{min} .*

Cette hypothèse est souvent vérifiée, il suffit par exemple qu'il existe une ressource qui croit strictement à chaque extension. Dans le chapitre 3, l'existence d'une telle ressource est vérifiée.

Comme pour l'algorithme 2, l'idée de base de l'algorithme de label-correcting est d'étendre de manière itérative les labels en disqualifiant les labels dominés. À chaque étape de l'algorithme nous disposons de $\mathcal{L}$, l'ensemble de tous les labels déjà créés et d'un tas Q contenant des labels qui ont été produit mais pas encore traités. On extrait de Q le label L_{min} défini par l'hypothèse 2.5.1. Nous l'étendons à tous les voisins atteignables tel que définis par 2.5.4. Pour chaque label L' ainsi généré, on vérifie qu'il n'existe pas un label de $L^* \in \mathcal{L}$ tel que $L^* \leq L'$. Si tel n'est pas le cas on ajoute L' à $\mathcal{L}$ et à Q et on supprime tous les labels de Q dominés par L' . L'algorithme se termine quand le tas Q est vide.

Proposition 2.5.3. *L'algorithme 3 calcule un plus court chemin multi-contraint de s à t , pour lequel il existe une relation d'ordre sur les labels et pour lequel les *REF* sont croissantes, en temps pseudo-polynomial*

$$O((|E(G)| + |V(G)|) \cdot W \log(|V(G)| \cdot W))$$

Algorithme 3: Plus court chemin multi-contraint de s à t

Données : $G = (V, E)$; $s, t \in V(G)$
Résultat : Un plus court chemin multi-contraint de s à t

```

1 Soit  $L_s$  le label du chemin  $p = \{s\}$  de longueur nulle ;
2  $\mathcal{L} = \{L_s\}$  ;
3  $Q.put(L_s)$  ;
4 tant que  $Q \neq \emptyset$  faire
5   Soit  $L_{min} \Leftarrow Q.pop()$  ;
6   Soit  $u$  le sommet de référence de  $L_{min}$  ;
7   pour tous les voisins  $v$  atteignables à partir de  $u$  faire
8     Soit  $L'$  le label  $L_{min}$  étendu par l'arc  $\{u, v\}$  ;
9     si  $\nexists L^* \in \mathcal{L}$  tel que  $L^* < L'$  alors
10        $Q.put(L')$  ;
11        $\mathcal{L} = L' \cup \mathcal{L}$  ;
12     fin
13     si  $\exists L^* \in Q$  tel que  $L' < L^*$  alors
14        $Q.remove(L^*)$  ;
15     fin
16   fin
17 fin
18 Renvoyer le chemin associé aux labels de  $t$  qui minimise la ressource  $r^*$ .
```

Sans suppression de label, l'Algorithme 3 considère tous les chemins de façon exhaustive et trouve donc le meilleur chemin multi-contraint. La proposition 2.5.2 montre que si L_1 est le label du début P_1 d'un chemin optimal alors en remplaçant P_1 par le chemin P_2 de L_2 , on obtient un autre chemin optimal. Par conséquent, le fait de supprimer un label dominé ne remet pas en cause l'optimalité de l'algorithme.

L'analyse de la complexité de l'algorithme est identique à celle de l'algorithme 2. Seul les labels non-dominés sont à prendre en compte, comme le montre la proposition 2.5.1. Pour chaque sommet, le nombre de chemins non-dominés est au plus W . Le label minimum L_{min} tiré du tas ne sera jamais dominé (par 2.5.1). Donc pour chaque sommet au plus $O(W)$ labels non-dominés sont à considérer. Chacun de ces labels est étendu un nombre de fois égal au degré sortant du sommet. Si l'on somme sur tous les sommets cela nous donne $O(|E| + |V|).W$ opérations d'extension de label. La taille du tas est dans le pire des cas de $O(|V|.W)$. Donc l'extraction du minimum se fait en $O(\log|V|.W)$.

Notes - État de l'art du plus court chemin multi-contraints Le problème du plus court chemin avec des contraintes de ressources est souvent utilisé dans la résolution des problèmes de tournées de véhicules et d'affectation stable d'équipes [17, 46, 50] comme un sous-problème pour la génération de colonnes. Le problème de plus court chemin avec deux contraintes de ressources a d'abord été étudié par Desrosiers [52, 51], qui le généralise plus tard au cas à plusieurs longueurs [53]. Par la suite, plusieurs variantes ont fait leur apparition dans la littérature. Par exemple une variante pour une métrique de temps dépendante linéairement des coûts des nœuds comme le propose [93] ou une variante qui se restreint aux chemins élémentaires [46]. En parallèle, les problèmes de plus court chemins avec des bornes maximales sur les pondérations autres que le coût ont également été étudiés. Le problème de plus court chemin avec poids contraint est un problème NP-difficile, même si l'on considère le cas où le graphe est acyclique et tous les poids et les coûts sont positifs [69]. Ce problème est un cas particulier du problème du plus court chemin avec fenêtres de temps, qui est lui même un cas particulier du problème de plus court chemin avec contraintes de ressources, qui utilise un vecteur de poids au lieu d'un seul. Il apparaît souvent comme sous-problème de génération de colonnes [17, 48, 82]. Les méthodes de résolutions de ce problème peuvent être regroupées en quatre catégories : celles fondées sur les k -plus courts chemins [83, 154], des méthodes de programmation dynamique [7, 49, 50, 51] utilisant des approches d'étiquetages (label) de sommets, des méthodes de relaxation lagrangienne de la contrainte de poids [18, 83, 103, 135, 154] et enfin des algorithmes d'approximations [84, 150].

Problème de déploiement FTTH

3

Résumé

Le problème du déploiement optimal d'un réseau de fibres optiques sous architecture de couplage passif consiste à concevoir un plan d'installation de fibres optiques de coût minimal permettant de connecter un ensemble de clients à un nœud de raccordement via un ensemble de coupleurs en respectant les contraintes techniques imposées par la norme. Nous présentons une modélisation de ce problème basée sur le concept de flot avec multiplicateurs et nous montrons comment obtenir par génération de colonnes une solution optimale fractionnaire du programme linéaire correspondant. Cette approche nécessite de résoudre à chaque itération un problème de plus court chemin généralisé multi-contraint. Pour cela, nous décrivons un algorithme pseudo-polynomial qui généralise au cas avec multiplicateurs l'algorithme de plus court chemin multi-contraint présenté dans [49]. Pour finir, nous détaillons une méthode d'arrondis de la solution fractionnaire et exposons les résultats expérimentaux obtenus avec cette approche.

3.1 Contexte du problème FTTH

Dans cette section nous décrivons le contexte qui a amené les opérateurs de télécommunications à adopter complètement la fibre optique comme moyen d'évolution de leurs réseaux. Ensuite, nous établissons un état de l'art des différentes modélisations proposées par la communauté scientifique dans le but de réduire efficacement le coût de déploiement de tels réseaux.

Contexte économique, technique et social

Depuis le début des années 2000, les offres internet à temps de connexion illimité ont considérablement modifié les habitudes de télécommunications de la population qui dès lors n'ont cessé d'évoluer. En effet, en plus de la consultation de pages internet et des échanges de fichiers, des sites d'hébergements de musiques et de vidéos ont fait leur apparition ces dernières années. Ils proposent une diffusion en Streaming de leurs contenus, qui envoient le média en flux continu à l'utilisateur en l'alimentant au fur et à mesure de sa progression. Cette solution permet de supprimer virtuellement le temps d'attente dû au téléchargement mais demande en revanche un débit suffisant du client. Dans le cas contraire, ce dernier recevra ce média de manière saccadé. De plus, dans un contexte de forte concurrence tarifaire, les Fournisseurs d'Accès Internet (FAI) ont également étendu leurs services en proposant par exemple la téléphonie, la télévision par internet et la vidéo à la demande. Ces nouvelles fonctionnalités sont intimement liées à l'augmentation de la bande passante qui résultait jusqu'à présent de l'adoption massive de la technologie de communication xDSL.

La technologie xDSL fournit un accès internet en transmettant des données numériques via le réseau téléphonique pré-existant. Le principe consiste à exploiter la partie non utilisée par la voix du spectre de fréquences disponible sur une ligne téléphonique. Pour des raisons économiques évidentes, cette technique permet de disposer d'un réseau sans les coûts d'installation de câbles jusqu'au client. Cependant, la consommation soutenue de médias digitaux et leurs échanges sont rapidement venus à bout des capacités de transit disponibles sur les câbles téléphoniques.

Pour faire face à la demande, des réseaux de nouvelles générations constitués de fibres optiques peuvent être déployés. Le réseau FTTH remplace intégralement le câble téléphonique en cuivre par de la fibre optique jusqu'au domicile du client. Plusieurs architectures permettent de définir un tel réseau. La plus simple consiste à raccorder directement chacun des clients. Il est également possible de déployer le réseau sous forme d'arbre comme le ferait un réseau de distribution d'eau ou d'électricité. Les méthodes d'évaluation des coûts [99, 28] et des études [110, 117] ont conduit les opérateurs à choisir l'architecture PON¹.

L'architecture PON permet à une fibre partant du nœud de raccordement optique de desservir plusieurs abonnés au réseau, grâce à des équipements réseaux optiques passifs appelés coupleurs. Deux principaux standards de normalisation ont été créés ; EPON standardisé par l'institut *IEEE* avec le 802.3ah groupe et l'institut *ITU* le

1. *Passive Optical Network* ou Point à multi-point passif

norme sous la référence $G.984.x$ et le nom de GPON.

Les récentes études sur l'évaluation des performances de ces deux types d'architectures [126, 113] mettent en avant la flexibilité d'évolution et le coût moindre d'une architecture GPON. Les contraintes techniques établies dans la version $G.984.1$ sont à prendre en compte dans le modèle théorique. Les contraintes de longueurs maximales de fibre et de temps maximal de propagation du signal sont couramment prises en compte par une contrainte d'affaiblissement maximal de $28dB$. Cela correspond à une distance maximale de $20km$ et un taux de division de 32. Le taux de division maximum recommandé est de 1 : 64, i.e. chaque fibre connectée au coupleur permet de raccorder au maximum 64 clients .

Cette architecture de réseaux optiques doit faire face à des problèmes de décision au niveau du déploiement et du dimensionnement du réseau afin de réduire au maximum les coûts tout en respectant la norme. Le problème que nous considérons peut être formulé de la façon suivante :

Problème: Déploiement FTTH	
Entrée:	Topologie du réseau, les coûts d'installations des fibres optiques et des coupleurs, des demandes de connexion des clients
Question:	Quel est le plan de déploiement de réseau qui minimise les coûts d'installation des fibres optiques et des coupleurs en respectant les contraintes techniques ?

État de l'art

Compte tenu du choix des normalisations à respecter et des contraintes techniques supplémentaires pouvant être rencontrées par les opérateurs, les précédentes études de ce problème ont abouti à différentes modélisations et méthodes de résolutions qui peuvent être regroupées de la manière suivante.

Heuristiques Une grande partie des méthodes de résolution sont basées sur des heuristiques, car le problème de déploiement de fibres optiques est un problème NP-difficile [123]. Dans cet article l'auteur modélise ce problème comme un problème de localisation. Il propose différentes heuristiques basées sur le calcul d'arbres couvrants, et sur un algorithme de regroupement. Pour décider la meilleure procédure à utiliser, un pré-traitement fondé sur la densité du graphe est appliqué.

Parmi les heuristiques proposées, certaines considèrent un espace métrique pour guider la construction de la topologie du réseau en restreignant les positions possibles des coupleurs [98]. Ces méthodes s'appuient sur le postulat que des clients qui habitent à proximité ont de fortes chances d'être raccordés à un même coupleur. Un algorithme de k -partitions découpe l'ensemble des clients en composantes dont la cardinalité permet de les connecter à un type de coupleurs. La position du coupleur au sein d'une composante peut être définie par un algorithme de Weiszfeld, qui détermine géométriquement le point médian d'un ensemble de points comme le propose [116]. Une fois la position des coupleurs fixée, il reste à résoudre le problème de raccordements des coupleurs entre eux. Dans [132, 112], les auteurs décrivent des algorithmes génétiques qui sont utilisés pour déployer le réseau à moindre coût, alors que [116] propose une procédure de recherche locale avec recuit simulé qui elle-même fait appel à une procédure qui résout un programme linéaire en nombres entiers (PLNE).

Programmation Linéaire en Nombres Entiers Parallèlement, des modélisations de ce problème en PLNE ont été suggérées. Cela permet d'obtenir une garantie d'optimalité mais demande en retour des techniques de résolutions autres qu'une simple utilisation d'un solveur de PLNE qui est vite limitée par la taille des instances, comme le montre l'article [30] qui compare l'efficacité d'une méthode de recherche locale tabou et la résolution du programme linéaire par un solveur commercial. Dans [95], une heuristique de partition génère un grand nombre d'ensembles de coupleurs pour chaque partition qui correspondent à une variable de décision du PLNE. La relaxation qui comporte un grand nombre de variables est résolue par un algorithme de génération de colonnes. Ce problème prend en compte une architecture sur deux niveaux de coupleurs. Pour modéliser virtuellement la contrainte d'éclatement, des modélisations de type hiérarchique avec plusieurs niveaux de coupleurs ont été proposées (modélisation k -niveaux). Le concept de base consiste à définir une relation d'ordre entre les coupleurs et à imposer une contrainte spécifiant que chaque coupleur doit être connecté à un seul coupleur de niveau immédiatement supérieur et peut desservir (être connecté à) plusieurs coupleurs de niveau immédiatement inférieur. Si on ajoute une contrainte sur le nombre maximum de niveaux traversés par une fibre, il est possible de prendre en compte dans une certaine mesure l'éclatement maximal en fixant pour chaque niveau un type de coupleurs comme proposé dans [31, 32]. Dans [80], une formulation en PLNE du problème de localisation d'usines qui utilise deux niveaux de coupleurs est comparée à une méthode de recherche

locale dans l'espace des solutions admissibles du problème. L'auteur souligne le fait que s'il fixe la position des coupleurs alors le problème revient à résoudre un problème de flot avec multiplicateurs en nombres entiers. Des travaux utilisant à la fois la position géographique des clients et les niveaux de coupleurs ont été effectués notamment dans [119] ; pour ces simulations l'auteur génère des grilles pour simuler des instances réalistes, ce que nous avons également fait pour nos simulations.

Instances De manière générale, les heuristiques permettent d'obtenir des résultats sur des grandes instances, mais la qualité de la solution n'est pas garantie. Celles utilisant un partitionnement du graphe, induisent implicitement que le raccordement entre deux éléments du réseau se fait en ligne droite sans respecter des conduits pré-existants, même si des auteurs prennent en compte des obstacles comme [112] en générant le graphe de visibilité. D'un autre côté, les méthodes exactes ne permettent de traiter que des petites instances, la contrainte d'intégrité étant pénalisante pour la résolution.

Positionnement et contribution

Nous proposons une modélisation du problème de déploiement de fibres optiques comme un problème de flot généralisé similaire à celle proposée par [80] en décrivant un PLNE très simple ayant peu de contraintes. Notre modèle, décrit dans [92], prend en compte à la fois le choix des coupleurs et le déploiement du réseau de fibres optiques. Nous proposons de résoudre sa relaxation linéaire par une technique de génération de colonnes qui a déjà été utilisée par [95] pour répondre à une problématique semblable. À notre connaissance, l'approche que nous proposons est la première à prendre en compte simultanément les contraintes de longueur de chemin, déjà abordées par [116, 30, 95], et d'éclatement maximal, prises en compte par [119, 80, 32]. Enfin, nous proposons une heuristique pour obtenir rapidement une solution entière de notre problème. Nous ne prenons pas en compte les problèmes liés au câble de fibres optiques. En effet il n'est pas envisageable de déployer physiquement les fibres une par une, ces dernières sont regroupées en câble de différents diamètres en fonction du nombre de fibres qu'ils contiennent. Cela fait apparaître des problèmes de fibres noires dus aux épissures sur un câble de fibre optique et le problème induit par les différents diamètres de câble à insérer dans un conduit pouvant être apparenté à un problème de circle-packing in circle [37, 144]. Nous ne prenons pas en compte les problèmes de fiabilité du réseau (en pratique ces problèmes ne sont pas pris en compte actuellement). Les précédentes études sur

le déploiement de réseau FTTH utilisent des méthodes en deux parties, dont la première consiste à positionner les coupleurs [95, 112, 116, 132]. Nous parvenons à nous affranchir de cette étape en reportant tous les coûts sur d'installations des coupleurs sur les liaisons comme décrit dans la section 3.2.

3.2 Modélisation du problème FTTH sous architecture PON

Dans cette section nous présentons une définition formelle du problème FTTH que nous allons considérer. Nous montrons ensuite que ce problème revient à résoudre “un problème de flot généralisé de coût minimum avec contraintes de longueurs sur les chemins”. Nous proposons ensuite une modélisation de ce problème sous la forme d'un programme linéaire en nombres entiers.

Définition du problème

Nous considérons un problème d'optimisation qui apparaît lorsqu'un opérateur de télécommunications souhaite déployer de manière efficace un réseau de fibres optiques pour connecter un ensemble de clients. Ce problème consiste à déterminer un plan d'installation de fibres optiques permettant de connecter chacun des clients à un nœud de raccordement que l'on nomme *OLT*². L'opérateur dispose de sites accueillants des coupleurs qui permettent d'éclater (répartir) le flux issu d'une fibre optique sur plusieurs fibres. Une étude de génie civile préalable permet de définir l'ensemble des liaisons inter-sites qui peuvent être équipées de fibres optiques. Pour chaque liaison, l'opérateur connaît : une capacité correspondant au nombre maximum de fibres qu'elle peut accueillir, un coût de déploiement d'une fibre optique dans celle-ci et l'affaiblissement subi par un signal la traversant en prenant en compte les épissures³.

Les coupleurs optiques sont des composants réseaux passifs disposant d'une entrée et de plusieurs sorties qui divisent un signal optique entrant sur ces sorties. On définit le facteur de concentration d'un coupleur comme son nombre de sorties, pour des raisons de production à grande échelle, les nombres de sorties sont des puissances de 2. Plus précisément, chaque coupleur est caractérisé par un entier $t \in \{1, 2, 3, 5\}$ et permet de réémettre le flux issu d'une fibre sur 2^t fibres. Cependant, une perte du signal est subie à chaque traversée d'un coupleur. Une borne théorique, proche des caractéristiques matérielles actuelles, donne $3dB$ par puissance de 2 du facteur de concentration, soit une perte de $3t dB$ pour un coupleur de type t . Il est possible

2. *Optical Line Termination*, parfois appelé NRO (Nœud de Raccordement Optique) dans la littérature

3. *Épissure* : raccord de deux fibres entre elles

3.2. MODÉLISATION DU PROBLÈME FTTH SOUS ARCHITECTURE PON

de reporter le coût d'un coupleur et l'affaiblissement qu'il engendre sur les liaisons sortantes. La figure 3.1 représente cette modélisation. On peut y voir la topologie du réseau avec les différents sites coupleurs. Pour chaque site coupleur on dispose de sommets représentant les différents types de coupleurs. Le coût du coupleur étant compris dans l'arc sortant de ces sommets, le fait de passer par celui-ci correspond à payer l'installation du coupleur. On relie tous les arcs entrants du site coupleur à un sommet que l'on relie aux sommets représentant les différents types de coupleurs par des arcs de coûts et d'affaiblissements nuls. Pour chaque arc sortant on fait correspondre un sommet, qui est relié aux sommets représentant les coupleurs par des arcs prenant en compte le coût, l'affaiblissement et l'éclatement du coupleur.

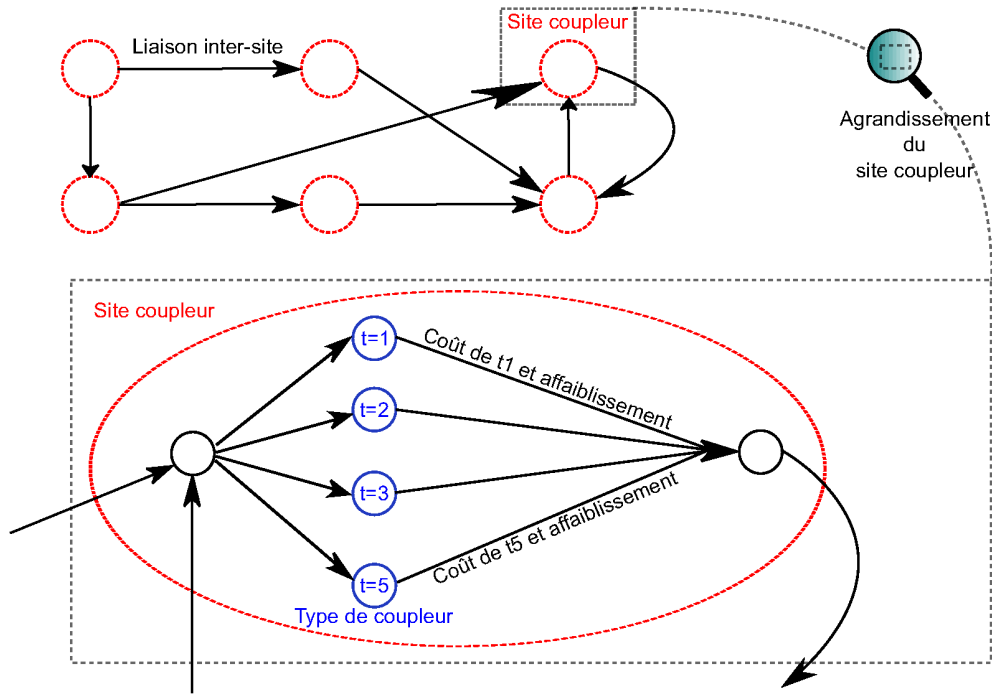


FIGURE 3.1 – Modélisation des coupleurs

Comme décrit dans la sous-section 3.1, nous nous basons sur une architecture réseau PON. Pour chaque chemin utilisé pour connecter un client à *OLT*, cette architecture impose que le produit des facteurs de concentration des arêtes du chemin soit inférieur ou égal à 64. Nous devons également pouvoir garantir que l'affaiblissement subi par le signal sur le chemin entre le client et *OLT* ne dépasse pas un certain seuil, noté NdB . La figure 3.2 offre une vue schématique de notre architecture. L'opérateur dispose d'un point de raccordement à son réseau mère nommé

OLT. Le déploiement de la fibre se fait via des tranchées ou des canalisations. Durant son trajet une fibre peut être éclatée plusieurs fois jusqu'à l'habitation du client. Le déploiement doit toujours respecter les contraintes techniques entre *OLT* et le client.

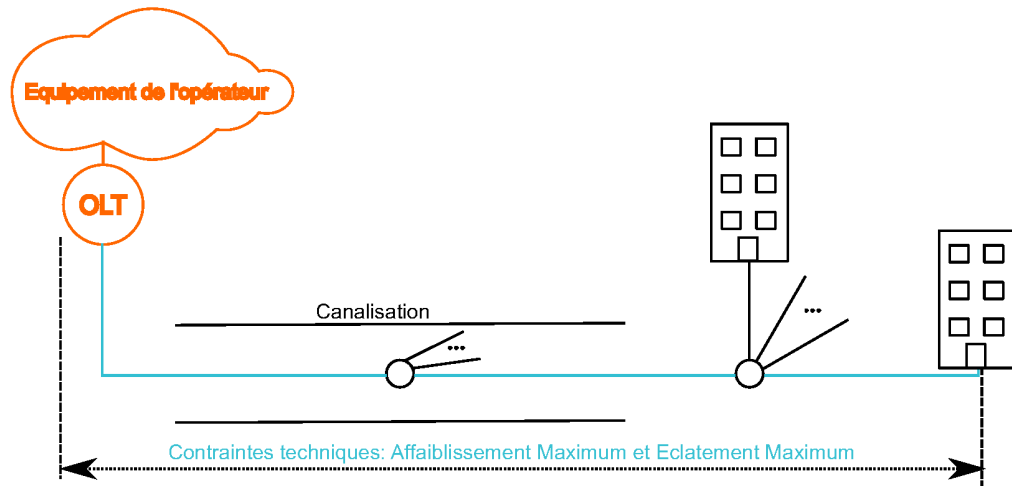


FIGURE 3.2 – Architecture PON

Modélisation en un problème de flot généralisé entier multi-contraint de coût minimum

Nous proposons d'utiliser la notion de flot avec multiplicateurs décrite dans la section 2.4 pour modéliser ce problème. Cette notion se distingue de la notion usuelle de flot par l'existence de multiplicateurs α pour chaque arc qui indique que si une unité de flot entre dans un arc alors α unités de flot en ressortent.

Cette modélisation devient naturelle si on considère le déploiement d'une fibre optique dans le réseau comme un flot, où une unité de flot entrant dans un arc représente le déploiement d'une fibre optique entre l'origine et l'extrémité de cet arc. L'existence des multiplicateurs permet de prendre en compte la présence des coupleurs en appliquant aux flots passant par un coupleur, un multiplicateur égal à l'inverse du facteur de concentration du coupleur, le flot est donc orienté des clients vers *OLT* comme représenté sur la figure 3.3.

Nous avons donc une modélisation qui pour une unité de flot entrant dans un arc représente le déploiement d'une fibre optique entre l'origine et l'extrémité de cet arc. Le coût des fibres est pris en compte dans le coût des arcs, il en va de même pour le coupleur dont le coût est répercuté sur l'arc sortant, c'est à dire pour chaque

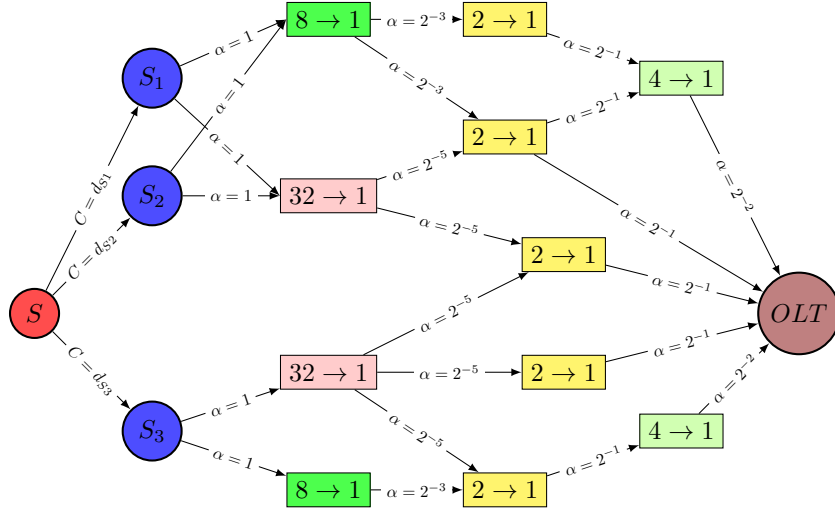


FIGURE 3.3 – Concentration du flot

unité de flot qui sort d'un coupleur son coût comprend le coût du coupleur plus les coûts relatifs aux fibres utilisant cet arc.

Pour que la modélisation soit valide, les chemins empruntés doivent prendre en considération les contraintes d'éclatement maximal et d'affaiblissement.

La contrainte d'intégrité du flot est dû à l'impossibilité de fractionner la demande d'un client sur plusieurs fibres optiques.

Nous devons donc fournir un dimensionnement de réseaux de coût minimum qui supporte un flot généralisé entier multi-contraint de coût minimum. Nous disposerons ainsi du plan de déploiement réseaux à effectuer et aussi du routage des communications à l'intérieur de ce dernier.

Modélisation des données sous forme de graphe

L'ensemble des liaisons sur lesquels des fibres optiques peuvent être déployées est représenté par un graphe orienté $G = (V, E)$.

L'ensemble des nœuds V comprend :

- les chambres logiques, qui regroupent des coupleurs de même type
- les clients, qui représentent physiquement une maison ou un bâtiment (dans ce dernier cas la demande du client correspond au nombre d'appartements à raccorder)
- une source S , qui est reliée à chaque client par un arc dont la capacité est égale aux nombres de fibres demandées par le client, de coût nul et de multiplicateur

unitaire

- OLT , qui est le sommet de convergence du flot

À chaque arc $e \in E$ sont assignées plusieurs pondérations :

- C_e une capacité, représentant le nombre de fibres que la liaison peut accueillir
- l_e un coût d'installation unitaire, qui prend en compte le coût d'installation d'un coupleur si l'arc sort d'un coupleur
- γ_e un affaiblissement linéique, qui tient compte de l'affaiblissement produit par un coupleur en l'ajoutant à l'arc y entrant
- α_e un multiplicateur, qui est égal 1 si l'origine de l'arc e n'est pas un coupleur et à l'inverse du facteur de concentration du coupleur sinon

Remarques Avec ce type de modélisation de données toutes les spécificités des coupleurs telles que le coût et le facteur de concentration sont reportés sur les arcs. Pour les demandes clients, nous effectuons une transformation classique d'un problème multi-puits multi-source en un problème de flot maximum. Soit $G = (V, E)$ un graphe avec un ensemble de sources $s = \{s_1, \dots, s_n\}$ et un ensemble de puits $t = \{t_1, \dots, t_m\}$ au lieu d'une seule source et un seul puits, nous voulons trouver le débit maximal à travers G tel que la d_i unités de flot atteignent t_i pour $i \in \{1, \dots, n\}$. Nous pouvons transformer le problème multi-puits multi-source dans un problème de flot maximum en ajoutant une source S et un puits T (couramment appelé super-source et super-puits). Pour $i \in \{1, \dots, n\}$, S est relié au sommet s_i et T est raccordé au sommet t_i par un arc de capacité d_i (voir la figure 3.4).

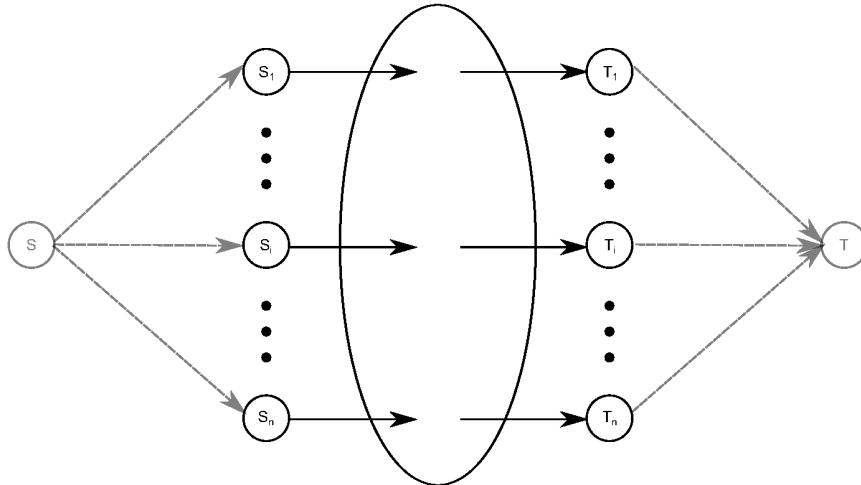


FIGURE 3.4 – Transformation multi-puits multi-sources en flot maximum

Modélisation en un programme linéaire en nombres entiers

Soit $\mathcal{P}$ l'ensemble des chemins allant de S à OLT et respectant les contraintes techniques. Un chemin est considéré comme une liste d'arcs comme décrit dans la section 2.1. Chaque chemin $p \in \mathcal{P}$ vérifie les contraintes techniques qui sont les suivantes :

1. le produit des facteurs de concentrations des coupleurs traversés par p est inférieur ou égal à 64
2. la somme des affaiblissements sur les arcs du chemin est inférieure ou égale à la valeur seuil Ndb

Une unité de flot entrant dans un arc représente le déploiement d'une fibre optique entre l'origine et l'extrémité de cet arc. Pour chaque chemin $p \in \mathcal{P}$, la variable de décision ϕ_p représente la quantité de flot envoyée par S à OLT via le chemin p .

Soient l_e et y_e , respectivement le coût de déploiement d'une fibre optique et la variable entière de décision de dimensionnement de l'arc e (combien de fibre vont être réellement déployées sur l'arc e). Pour chaque arc $e_q \in p := (e_1, e_2, \dots, e_r)$, on note $x_p(e_q) := \prod_{i=1}^q \alpha_{e_i}$ la quantité de flot qui sort de l'arc e_q lorsqu'une unité de flot est émise à partir de S et transite sur les arcs $e_1, e_2, \dots, e_{q-1}$.

Nous proposons le programme linéaire en nombres entiers $FTTH_{Int}$ pour modéliser notre problème de dimensionnement de réseaux de coût minimum supportant un flot généralisé entier multi-contraint.

On note $d := \sum_{k \in \mathcal{K}} d_k$ le flot (nombre total de fibres) demandé par les clients.

$$FTTH_{Int} = \left\{ \begin{array}{ll} \text{Min} & \sum_{e \in E} y_e l_e \\ \text{s.c} & \\ & \sum_{p \in \mathcal{P}} \phi_p = d \quad (1) \\ & \sum_{e \in p} x_p(e) \phi_p \leq y_e, \forall e \in E \quad (2) \\ & y_e \leq C_e, \forall e \in E \quad (3) \\ & \phi_p \in \mathbb{N}, \forall p \in \mathcal{P} \\ & y_e \in \mathbb{N}, \forall e \in E \end{array} \right.$$

Description de $FTTH_{Int}$ La fonction objectif consiste à minimiser le coût du dimensionnement du réseau de fibres optiques. La contrainte (1) assure que chaque client reçoit le nombre de fibres optiques demandées. La quantité $x_p(e) \phi_p$ représente le

nombre de fibres optiques requis sur l'arc e par l'émission à partir de S de ϕ_p unités de flot sur le chemin p . Les contraintes (2) certifient que le dimensionnement du réseau supporte le routage des demandes. Les contraintes (3) assurent que le nombre de fibres déployées sur chaque liaison e ne dépasse pas le nombre de fibres qu'elle peut accueillir. Les contraintes techniques sont prises en compte implicitement dans la définition de l'ensemble $\mathcal{P}$.

Nous proposons maintenant le programme linéaire $FTTH$ pour modéliser notre problème de flot généralisé multi-contraints de coût minimum induit dans le précédent programme linéaire en nombres entiers. La relaxation linéaire est obtenue en remplaçant chaque contrainte d'intégrité $\phi_p \in \mathbb{N}$ par $\phi_p \geq 0$. Par la suite, nous nous intéressons particulièrement à la résolution optimale de cette relaxation linéaire.

$$FTTH = \begin{cases} \text{Min} & \sum_{p \in \mathcal{P}} \sum_{e \in p} l_e x_p(e) \phi_p \\ \text{s.c} & \sum_{p \in \mathcal{P}} \phi_p = d & : \lambda & (4) \\ & \sum_{p \in \mathcal{P}: e \in p} x_p(e) \phi_p \leq C_e, \forall e \in E & : \beta_e & (5) \\ & \phi_p \geq 0, \forall p \in \mathcal{P} \end{cases}$$

Nous remarquons que la variable y_e présente dans $FTTH_{Int}$ a disparu, car elle correspond au nombre de fibres (un entier) devant être réellement déployées sur l'arc e . $FTTH$ étant la relaxation des contraintes d'intégrités de $FTTH_{Int}$, les variables y_e y sont inutiles.

Nous considérons le programme linéaire dual de $FTTH$ portant sur les variables duales λ et β_e associées aux contraintes (4) et (5) :

$$FTTH^* = \begin{cases} \text{Max} & \lambda d - \sum_{e \in E} C_e \beta_e \\ \text{s.c} & \lambda - \sum_{e \in p} x_p(e) \beta_e \leq \sum_{e \in p} x_p(e) l_e, \forall p \in \mathcal{P} : \phi_p & (6) \\ & \beta_e \geq 0, \forall e \in E \end{cases}$$

La contrainte (6) peut-être réécrite $\sum_{e \in p} x_p(e) (\beta_e + l_e) \geq \lambda$. Cette écriture de la contrainte (6) sera reprise dans la section suivante pour la mise en place de l'algorithme de résolution du problème $FTTH$.

3.3 Résolution de la relaxation linéaire

Dans cette section, nous décrivons la méthode utilisée pour résoudre le programme linéaire $FTTH$ dont le grand nombre de variables rend la résolution difficile. Nous mettons en place un algorithme de génération de colonnes pour parvenir à résoudre $FTTH$. À chaque itération de notre algorithme, nous faisons appel à un second algorithme de séparation (cf section 2.3). Nous présentons ce dernier ainsi qu'une version modifiée qui utilise des informations qui sont calculées lors de la résolution du sous-problème de séparation. Nous montrons ici comment calculer efficacement une solution optimale de cette relaxation en identifiant un nombre réduit de chemins qui supportent une solution optimale.

Génération de colonnes

Le modèle $FTTH$ introduit une variable ϕ_p pour chaque chemin $p \in \mathcal{P}$ qui vérifie les contraintes techniques. Ce nombre de variables est en général très grand et non-polynomial par rapport à la taille du réseau. Pour résoudre ce problème nous appliquons la méthode de génération de colonnes présentée dans la section 2.3.

Nous allons résoudre une succession de programmes linéaires $FTTH(\bar{\mathcal{P}})$, portant sur les variables associées à un sous-ensemble de chemins $\bar{\mathcal{P}} \subseteq \mathcal{P}$. À chaque itération, la solution optimale duale de $FTTH(\bar{\mathcal{P}})$ permet d'identifier un nouveau chemin dont l'ajout à $\bar{\mathcal{P}}$ améliore le plan de déploiement courant.

Pour mettre en évidence un critère permettant d'identifier ces chemins, on peut considérer $(\lambda, \beta_e : e \in E)$ une solution optimale de $FTTH^*(\bar{\mathcal{P}})$ le programme linéaire dual de $FTTH(\bar{\mathcal{P}})$. Trouver un chemin $p \in \mathcal{P} \setminus \bar{\mathcal{P}}$ tel que la solution optimale de $FTTH(\mathcal{P} \cup p)$ soit meilleure que celle $FTTH(\mathcal{P})$ revient à identifier une contrainte du dual $FTTH^*(\mathcal{P})$ violée par la solution duale $(\lambda, \beta_e : e \in E)$, c'est à dire un chemin $p \in \mathcal{P}$ tel que $\sum_{e \in p} x_p(e)(\beta_e + l_e) < \lambda$.

Soit $dist(e) := (\beta_e + l_e)$ le coût de transit d'une unité de flot entrant dans l'arc e . Tout chemin respectant les contraintes techniques qui permet de faire transiter une unité de flot de S à OLT avec un coût inférieur à λ identifie une contrainte duale violée. Pour savoir si un tel chemin existe, on doit résoudre un *problème de plus court chemin généralisé multi-contraint* qui revient à trouver parmi l'ensemble des chemins respectant les contraintes techniques, un plus court chemin généralisé p qui permet de faire transiter une unité de flot à moindre coût en prenant $dist(e)$ comme coût de l'arc e .

Si le chemin p calculé a un coût $\sum_{e \in p} x_p(e) \text{dist}(e)$ strictement inférieur à λ , on l'ajoute à $\mathcal{P}$ et on réitère. Sinon, la solution duale courante $(\lambda, \beta_e : e \in E)$ du programme linéaire $FTTH^*(\overline{\mathcal{P}})$ est également valide pour $FTTH^*(\mathcal{P})$ et donc également optimale pour ce problème. Par dualité, la solution primale courante de $FTTH(\overline{\mathcal{P}})$ est également optimale pour le programme linéaire complet $FTTH = FTTH(\mathcal{P})$ justifiant ainsi la proposition suivante.

Proposition 3.3.1. *L'algorithme 4 calcule une solution optimale de la relaxation linéaire du problème FTTH.*

Algorithme 4: Résolution de $FTTH$ par génération de colonnes	
Données : $G = (V, E)$; $(s_k, d_k) \in V(G) \times \mathbb{R}_+$ pour $k \in \mathcal{K}$; $OLT \in V(G)$; $l \in \mathbb{R}_+^{E(G)}$; $C, 1/\alpha, \gamma \in \mathbb{N}^{E(G)}$	
Résultat : $(\phi_p : p \in \mathcal{P})$ une solution optimale de $FTTH$	
1	$\overline{\mathcal{P}} \leftarrow$ un ensemble arbitraire de chemins valides;
2	répéter
3	Résoudre le programme linéaire dual $FTTH^*(\overline{\mathcal{P}})$;
4	Soit $(\lambda, \beta_e : e \in E)$ une solution optimale $FTTH^*(\overline{\mathcal{P}})$;
5	Calculer un plus court chemin généralisé $p \in \mathcal{P}$ avec $\text{dist}(e) = (\beta_e + l_e)$ comme coût de l'arc e ;
6	si $\sum_{e \in p} x_p(e) \text{dist}(e) < \lambda$ alors
7	$\overline{\mathcal{P}} \leftarrow \overline{\mathcal{P}} \cup \{p\}$;
8	fin
9	jusqu'à $\sum_{e \in p} x_p(e) \text{dist}(e) \geq \lambda$;
10	Renvoyer la solution optimale $(\phi_p : p \in \mathcal{P})$ de $FTTH(\overline{\mathcal{P}})$;

La recherche d'un plus court chemin généralisé est exécutée un très grand nombre de fois. Son efficacité est donc un élément déterminant pour la mise en œuvre de cette méthode. La complexité de l'algorithme que nous proposons dans la section suivante, dépend de la grandeur des nombres des ressources d'affaiblissement et de concentration. Dans notre cas, seul la ressource de coût, qui est calculée en fonction des variables duales, varie. Donc notre algorithme de plus court chemin généralisé a toujours les mêmes paramètres de complexité. Par conséquent notre méthode de génération de colonnes a cette même complexité (cf. section 2.3).

Oracle de séparation : plus court chemin généralisé multi-contraint

En se basant sur le critère de Dantzig, une méthode efficace de génération de colonnes implique, pour notre problème de minimisation, l'identification d'une variable hors base dont le coût réduit négatif soit minimum. C'est pourquoi nous résolvons un problème de plus court chemin généralisé multi-contraint pour sélectionner les chemins dont l'utilisation permet d'améliorer la solution courante.

Le problème du plus court chemin généralisé multi-contraint est NP-Complet, car on peut réduire le problème de plus court chemin avec deux pondérations à ce problème comme décrit dans la section 18.

L'algorithme pseudo-polynomial que nous proposons pour résoudre ce problème est basé sur la programmation dynamique au même titre que l'algorithme de label-correcting classique décrit dans la section 18.

La programmation dynamique, contrairement aux heuristiques et méthode de relaxation lagrangienne aussi utilisées pour résoudre ce type de problème, offre une résolution exacte qui, dans notre cas, donne un certificat d'optimalité pour la résolution du problème *FTTH*.

Notre algorithme utilise des *labels* qui définissent un chemin entre S et un sommet du graphe. Chaque label L est constitué de cinq attributs qui ensemble caractérisent le chemin en construction :

- $u(L)$, le sommet auquel le label L fait référence et qui est l'extrémité du chemin qu'il représente
- $prec(L)$, le label à partir duquel L a été étendu
- $c(L)$, le coût du chemin associé au label L
- $\Gamma(L)$, l'affaiblissement théorique subi par un signal empruntant ce chemin
- $\tau(L)$, le facteur de concentration du chemin, (l'équivalent de $x_p(e)$ dans le problème *FTTH*)

On définit L' l'*extension* du label L par l'arc uv en affectant à L' les attributs suivant :

- $u(L') = v$
- $prec(L') = L$
- $c(L') = c(L) + dist(uv)\tau(L')$, la fonction de coût du chemin faisant référence à la contrainte (6) du problème *FTTH**
- $\Gamma(L') = \Gamma(L) + \gamma_{uv}$, l'affaiblissement du chemin étant la somme des affaiblissements des arcs qu'il utilise

- $\tau(L') = \tau(L)/\alpha_{uv}$, la concentration étant le produit des facteurs de concentration des arcs du chemin

On utilise une relation de domination entre les labels qui définit un ordre partiel. Un label L_1 *domine* un label L_2 si :

- $u(L_1) = u(L_2)$, ils référencent le même sommet
- $\tau(L_1) \leq \tau(L_2)$, ils correspondent à des chemins ayant une concentration inférieur ou égal
- $c(L_1) \leq c(L_2)$, le coût du chemin associé à L_1 est inférieur ou égal à celui du chemin représenté par L_2
- $\Gamma(L_1) \leq \Gamma(L_2)$, l'affaiblissement de L_1 est inférieur ou égal à celui de L_2

Si le label L_1 domine L_2 , en remplaçant le sous-chemin représenté par L_2 par celui de L_1 dans un chemin entre S et OLT , le chemin obtenu est au moins aussi bon sur tous les critères. Par conséquent, il est inutile de conserver L_2 .

L'algorithme que nous proposons utilise une liste pour stocker l'ensemble des labels correspondant aux labels devant être traités à la prochaine itération. Initialement, cette liste contient un seul label L_S qui correspond au chemin trivial réduit au seul sommet S dont les attributs sont : $u(L_S) = S$, $prec(L_S) = \emptyset$, $\tau(L_S) = 1$, $c(L_S) = 0$, $\Gamma(L_S) = 0$. Tant que la liste n'est pas vide, à chaque itération de l'algorithme, on retire le label L situé en tête de la liste. On parcourt l'ensemble des voisins v de $u = u(L)$, on étend L en v et on ajoute le label obtenu à la liste s'il est valide.

Un label est *valide* s'il respecte les contraintes techniques décrites dans la section 3.2 et s'il n'est pas dominé. Plus formellement un label L' est valide, si les trois conditions suivantes sont vérifiées :

- L' n'est dominé par aucun autre label L''
- $\Gamma(L') \leq NdB$, le chemin respecte l'affaiblissement maximal
- $1/\tau(L') \leq 64$, le chemin respecte l'éclatement maximal

À la fin du processus de marquage, si OLT n'a reçu aucun label, il n'existe aucun chemin respectant les contraintes techniques entre S et OLT . Dans le cas contraire, le label L^* de OLT dont l'attribut coût est minimum correspond à un plus court chemin généralisé multi-contraint.

La correction de cet algorithme est basée sur les deux observations suivantes. D'une part, si on omet la condition de non-domination, l'algorithme considère tous les chemins élémentaires entre S et OLT respectant les contraintes de concentration

et d'affaiblissement. D'autre part, ignorer les labels dominés n'élimine aucun chemin optimal car dans le cas contraire, échanger la partie dominé du chemin optimal produira un chemin encore meilleur, ce qui contredit son optimalité.

Remarque La ressource Γ est strictement positive. Sur le chemin cette ressource est strictement croissante. Notre algorithme choisit parmi les chemins en cours de calcul celui dont la ressource Γ est minimum et dans le cas où plusieurs chemins aurait la même valeur, on privilégie celui dont la ressource τ est minimum, comme décrit dans la section 18. Ceci permet de garantir qu'un label L_{min} tiré de la liste représentant un chemin atteignant le sommet v restera non-dominé de façon permanente.

La complexité de l'algorithme se base sur le fait que chaque sommet reçoit au plus $NdB \times \log_2 64$ labels non-dominés et chacun de ces labels est considéré un nombre de fois proportionnel au degré sortant du sommet. Un simple tas binaire permet d'obtenir le plus petit élément de la liste par l'opération *pop* et d'ajouter un élément avec l'opération *push* en temps logarithmique par rapport à la taille de la file qui dans le pire des cas est $6 \times NdB \times |V(G)|$.

Cet algorithme s'exécute donc en temps pseudo-polynomial

$$O((|E(G)| + |V(G)|) \times 6 \times Ndb) \times \log(6 \times NdB \times |V(G)|)$$

Proposition 3.3.2. *L'algorithme 5 calcule en temps pseudo-polynomial $O((|E(G)| + |V(G)|) \times 6 \times Ndb) \times \log(6 \times NdB \times |V(G)|)$ un plus court chemin généralisé multi-constraint entre S et OLT .*

Algorithme 5: Plus court chemin généralisé multi-contraint de S à OLT

Données : $G = (V, E)$; $S, OLT \in V(G)$; $dist \in \mathbb{R}_+^{E(G)}$; $\alpha, \gamma \in \mathbb{N}^{E(G)}$

Résultat : Un plus court chemin généralisé multi-contraint de S à OLT

```

1  $Q.put(L_s)$  avec  $u(L_s) = s$ ,  $prec(L_s) = \emptyset$ ,  $\tau(L_s) = 1$ ,  $c(L_s) = 0$ ,  $\Gamma(L_s) = 0$  ;
2 tant que  $Q \neq \emptyset$  faire
3    $L \leftarrow Q.pop()$  // Extrait le plus petit élément de la pile;
4   pour tous les voisins  $v$  de  $u(L)$  dont un label  $L'$  peut être étendu à partir
     de  $L$  faire
5      $Q.put(L')$  avec  $u(L') = v$ ,  $prec(L') = L$ ,  $c(L') = c(L) + dist(uv) \times \tau(L')$ ,
6      $\Gamma(L') = \Gamma(L) + \gamma_{uv}$  et  $\tau(L') = \tau(L)/\alpha_{uv}$ ;
7      $Q.remove(L'')$ , pour tout  $L''$  dominé par  $L'$  ;
8   fin
9 fin
10 Renvoyer le chemin associé au label  $L^*$  de  $OLT$  dont l'attribut coût est
    minimum.
```

Remarque et modifications de l'algorithme

Plusieurs méthodes de résolution ont été proposées pour le calcul de plus courts chemins multi-contraints, tel que des algorithmes d'approximations, des méthodes de relaxation Lagrangienne et la programmation par contraintes. Nous avons choisi de tirer partie de la programmation dynamique pour la génération de colonnes, car à la fin de l'exécution de l'algorithme 5, plusieurs chemins non-dominés peuvent être calculés. Nous proposons donc de les utiliser en les ajoutant s'ils correspondent à une contrainte du dual $FTTH^*$ violée.

Une autre variante que nous proposons est celle de la recherche du chemin dans le sens de OLT vers S . Ce sens de recherche implique une modification de la fonction du coût du chemin. Lors de l'extension d'un label L' à partir de L via l'arc uv la mise à jour du coût est donnée par la formule suivante :

$$c(L') = \frac{c(L) + dist(uv)}{\alpha_{uv}}$$

En effet, si une unité de flot en $u(L)$ coûte $c(L)$ alors on peut faire arriver α_{uv} unité de flot en v avec un coût $c(L) + dist(uv)$. En mettant à l'échelle, on obtient une unité de flot en v avec un coût de $(c(L) + dist(uv))/\alpha_{uv}$, comme décrit dans [61].

Pour nos simulations nous disposons donc de quatre variantes de génération de chemins que l'on nomme de la manière suivante :

3.4. HEURISTIQUE DE GÉNÉRATION D'UNE SOLUTION ENTIÈRE

1. F , la première méthode qui consiste à ajouter le chemin qui viole le plus les contraintes du programme linéaire dual $FTTH^*$ et qui respecte les contraintes techniques
2. FM , qui consiste à ajouter tous les chemins violant les contraintes de $FTTH^*$ calculés lors de l'exécution de l'algorithme de label-correcting via une recherche de S à OLT
3. B , qui, comme pour F , ajoute le chemin dont le coût marginal est minimum et qui respecte bien sûr les contraintes techniques ; mais en effectuant une recherche de OLT vers S en utilisant la fonction de coût explicitée précédemment
4. BM , qui ajoute tous les plus courts chemins calculés lors du bon déroulement de l'algorithme de programmation dynamique

3.4 Heuristique de génération d'une solution entière

Dans cette section, nous décrivons une méthode pour calculer une solution entière au problème $FTTH_{int}$ en résolvant une séquence de relaxations linéaires et en arrondissant les solutions fractionnaires entières de ces programmes linéaires.

À chaque étape, sur la base de la relaxation linéaire courante, l'algorithme choisit un arc et fixe le nombre de fibres qui vont être déployées sur ce lien. Puis il met à jour la relaxation en fonction de cette décision et passe à l'itération suivante. Ce processus se termine lorsque le déploiement courant satisfait les demandes de connexion des clients.

Soit E' l'ensemble des arcs pour lequel le nombre de fibres a déjà été fixé au début de la $i^{\text{ème}}$ itération. Soit f_e le nombre de fibres installées sur l'arc $e \in E'$ et y_e le nombre de fibres requis pour satisfaire la demande du client. Nous désignons par $FTTH'(f_e : e \in E)$ le programme linéaire suivant qui prend en compte la solution partielle courante ($f_e : e \in E'$) et minimise le coût de déploiement des fibres supplémentaires nécessaires pour satisfaire la demande en fibres des clients.

$$FTTH'(f_e : e \in E) = \left\{ \begin{array}{l} \text{Min } \sum_{p \in \mathcal{P}} \sum_{e \in p} l_e y_e \\ \text{s.c.} \\ \sum_{p \in \mathcal{P}} \phi_p = d \\ \sum_{p \in \mathcal{P} : e \in p} x_p(e) \phi_p \leq y_e, \forall e \in E \\ \phi_p \in \mathbb{R}, \forall p \in \mathcal{P} \\ \\ y_e = f_e \forall e \in E' \\ 0 \leq y_e \leq C_e \forall e \in E - E' \end{array} \right.$$

Proposition 3.4.1. *L'algorithme 6 calcule une solution entière réalisable du problème $FTTH_{int}$.*

Algorithme 6: Procédure d'arrondis

Données : $G = (V, E)$; $S, OLT \in V(G)$; $dist \in \mathbb{R}_+^{E(G)}$; $\alpha, \gamma \in \mathbb{N}_+^{E(G)}$
Résultat : $(y_e : e \in E)$ est une solution entière du problème $FTTH_{int}$

- 1 $E' = \emptyset$;
- 2 **tant que** $\exists y_e$ fractionnaire **faire**
 - 3 Chercher une solution optimale $(\phi_p : p \in \mathcal{P})$ de $FTTH'(f_e : e \in E')$;
 - 4 Soit e' un arc avec un nombre fractionnaire de fibres;
 - 5 $y_{e'} := \lceil y_{e'} \rceil$;
 - 6 $E' = E' \cup \{e'\}$;
- 7 **fin**
- 8 Retourner $(y_e : e \in E)$.

La solution générée par cette procédure offre une solution réalisable au problème $FTTH_{int}$ justifiant la correction de l'algorithme. En effet, lorsque l'algorithme 6 se termine, l'ensemble des variables y_e sont entières et le graphe support G_y des variables y_e , où y_e représente la capacité de l'arc e , autorise une solution du problème de flots généralisés dont l'ensemble des chemins respectent les contraintes techniques. En d'autres termes, on peut dire que l'on déploie à travers les liaisons physiques du réseau de l'opérateur un nombre entier de fibres optiques permettant de satisfaire la demande des clients en respectant les contraintes imposées par la norme GPON.

La procédure proposée construit une solution en au plus $O(|E|)$ itérations. À chaque étape, un arc e , sur lequel le nombre de fibres à installer est fractionnaire, est choisi et on fixe sa capacité à l'entier supérieur. Les capacités des arcs étant

entières le problème reste réalisable.

L'algorithme 6 ne garantit pas une solution dans laquelle les ϕ_p sont entiers, mais la forme des instances considérées lors de nos simulations avec ses contraintes de capacités, nous ont donné dans la majorité des cas des solutions satisfaisantes. Pour obtenir une solution exacte au problème $FTTH_{int}$ un algorithme de type "Branch and Price" doit être envisagé.

3.5 Résultats des simulations

Dans cette section, nous présentons les simulations effectués et nous comparons les différentes stratégies de génération de colonnes. Les résultats sont organisés en deux parties, la première porte sur l'algorithme de résolution du programme linéaire $FTTH$ et la seconde concerne l'heuristique d'arrondis pour générer une solution entière du problème $FTTH_{int}$.

L'implémentation de nos algorithmes a été faite en JAVA, pour résoudre les programmes linéaires, nous utilisons le solveur libre GLPK⁴. Les tests ont été effectués sur une machine disposant d'un processeur Intel Core2 Duo cadencé à 2Ghz.

Instances générées

La stratégie utilisée par les opérateurs consiste à utiliser en priorité leurs infrastructures. Si cela s'avère insuffisant ils privilégient des canalisations pré-existantes. Les problèmes d'optimisation du déploiement de fibres optiques se posent essentiellement dans des zones très denses en population. Les réseaux de type *Manhattan* (grille carrée) permettent de modéliser assez fidèlement ce type de région. C'est pourquoi la génération d'instances sur une grille comme le propose [119] nous a semblé être appropriée. À notre connaissance l'estimation du budget réalisé par l'opérateur se fait de la manière suivante ; on estime la distance moyenne d'un client par rapport au nœud de raccordement, que l'on multiplie par le nombre total de client à raccorder pour avoir la distance totale de câble à déployer et pour finir on multiplie cette longueur par le coût linéaire moyen d'une pose de câble de fibre optique. Aucune rétro-ingénierie (post-déploiement) n'est appliquée à notre connaissance pour valider ce budget.

Afin d'évaluer l'efficacité de notre méthode de résolution du problème $FTTH$, nous avons résolu plusieurs cas générés aléatoirement du problème avec différentes stratégies pour la génération de chemins.

4. version 4.45 : <http://www.gnu.org/software/glpk/>

CHAPITRE 3. PROBLÈME DE DÉPLOIEMENT FTTH

Les instances qui ont permis d'effectuer ces simulations ont été générées aléatoirement de la manière suivante. Considérons une grille carrée de taille $(n \times n)$ où n varie entre 11 et 22. Le nombre de sites clients est une puissance de 2 et chaque site client requiert une fibre optique. Les quatre types de coupleurs $t \in \{1, 2, 3, 5\}$ sont répartis également de façon uniformément aléatoire sur la grille. Les sites clients et les coupleurs sont également distribués de façon aléatoire sur la grille. Les coûts, l'affaiblissement et la capacité des arcs sont compris dans un intervalle de valeurs et choisis de manière aléatoire en utilisant une distribution uniforme.

Le tableau 3.1 résume les tailles des instances générées.

Taille	11	12	13	14	15	16	17	18	19	20	21	22
$ V $	122	145	170	197	226	257	290	325	362	401	442	485
$ E $	252	296	344	396	452	512	576	644	716	792	872	956

TABLE 3.1 – Caractéristiques des instances

La méthode BigM, qui consiste à créer une colonne artificielle réalisable de coût très grand, est utilisée pour l'initialisation de l'algorithme de génération de colonnes. Cela consiste à ajouter une variable artificiellement créée pour garantir une solution réalisable lors de la première itération de l'algorithme de génération de colonnes. On attribue un coût suffisamment grand à cette variable pour être sûr qu'elle sera nulle dans une solution optimale si il existe une solution réalisable du problème initial.

Nos expériences montrent que la résolution du problème de plus court chemin généralisé multi-contraint représente plus de 98.9% du temps de calcul.

Nous mettons en œuvre quatre stratégies pour la génération des chemins décrites dans la section 10. Nous proposons les notations suivantes :

- La stratégie F utilise l'algorithme 5 tel qu'il est présenté
- La stratégie FM recycle les chemins non-dominés calculés lors de la recherche du plus court chemin
- La stratégie B effectue une recherche de *OLT* vers *S*
- La stratégie BM effectue une recherche de *OLT* vers *S* et recycle les chemins non-dominés calculés lors de la recherche du chemin

Chaque méthode retourne une solution optimale.

Le tableau 3.2 synthétise les expérimentations de résolution du problème *FTTH* effectuées. Il existe quatre axes d'évolution de l'instance : la taille de la grille, le

nombre de sites clients, les quatre méthodes de génération de chemins et le nombre de coupleurs.

Une instance est caractérisée par les paramètres suivant :

- le nombre de site clients qui est un multiple de 64 varie de 64 à 256, il est limité par la taille de l'instance ;
- la taille de la grille varie dans l'intervalle $[11, 22]$;
- la méthode de génération de chemins utilisée.

Le nombre de coupleurs n'est pas explicite dans ce tableau car pour chaque caractéristique d'instance représentée, nous effectuons 3 tirages ayant respectivement 12, 24 et 36 sites coupleurs. Chaque résultat représente la moyenne de 20 instances de même caractéristiques. Pour chaque instance caractéristique nous fournissons trois indicateurs :

- *cpu*, qui indique le temps CPU utilisé en moyenne pour trouver une solution optimale
- *chemins*, la cardinalité du sous-ensemble de chemins utilisés en moyenne pour définir un ensemble de chemins qui supportent la solution optimale
- *itérations*, qui mesure le nombre d'itérations en moyenne qu'il faut à l'algorithme de génération de colonnes pour renvoyer une solution optimale

Nous considérons comme référence la série de tests comprenant 64 sites clients et utilisant la méthode F de génération de chemins. Ainsi nous pouvons évaluer l'algorithme que nous proposons en fonction des méthodes de génération de chemins utilisées, mais aussi en fonction du nombre de sites clients. Pour mettre en valeur ces résultats nous résumons pour chaque agrégation de tuple de données (méthode, nombre de clients, indicateur) un pourcentage par rapport à la référence.

Taille de la grille	F : Source $\rightsquigarrow$ Puits			B : Puits $\rightsquigarrow$ Source			FM : F + chemins recyclés			BM : B + chemins recyclés		
	cpu	chemins	itérations	cpu	chemins	itérations	cpu	chemins	itérations	cpu	chemins	itérations
64 clients												
11	0.5	64	64	0.8	64	64	0.2	255	25	0.3	256	26
12	0.7	64	64	1.1	64	64	0.3	255	25	0.4	255	25
13	1.0	64	64	1.5	64	64	0.4	256	26	0.6	256	26
14	1.3	64	64	2.0	64	64	0.5	256	26	0.8	256	26
15	1.7	64	64	2.6	64	64	0.7	256	26	1.0	256	26
16	2.2	64	64	3.4	64	64	0.9	257	26	1.4	257	27
17	2.7	64	64	4.3	64	64	1.1	257	26	1.7	257	27
18	3.5	64	64	5.5	64	64	1.4	256	26	2.2	256	26
19	4.4	64	64	6.9	64	64	1.8	257	27	2.7	257	27
20	5.4	64	64	8.6	64	64	2.2	257	27	3.3	256	26
21	6.5	64	64	10.2	64	64	2.6	257	26	4.0	257	27
22	7.9	64	64	12.5	64	64	3.2	257	27	4.9	256	26
-	100%	100%	100%	152%	100%	100%	41%	400%	41%	62%	400%	41%
128 clients												
14	2.6	128	128	4.0	128	128	1.0	512	52	1.6	512	52
15	3.4	128	128	5.2	128	128	1.4	511	51	2.0	512	52
16	4.4	128	128	6.8	128	128	1.8	512	52	2.8	512	52
17	5.4	128	128	8.6	128	128	2.2	513	53	3.4	512	52
18	7.0	128	128	11.0	128	128	2.9	512	52	4.4	512	52
19	8.8	128	128	13.8	128	128	3.6	512	52	5.4	512	52
20	10.8	128	128	17.2	128	128	4.4	511	51	6.6	511	51
21	13.0	128	128	22.4	128	128	5.2	512	52	8.0	512	52
22	15.8	128	128	25.0	128	128	6.4	512	52	9.8	512	52
-	200%	200%	200%	304%	200%	200%	82%	800%	81%	124%	800%	81%
256 clients												
18	14.0	256	256	22.0	256	256	5.6	1024	105	8.8	1024	105
19	17.6	256	256	27.6	256	256	7.2	1024	105	10.8	1024	105
20	21.6	256	256	34.4	256	256	8.8	1023	104	13.2	1023	104
21	26.0	256	256	40.8	256	256	10.4	1024	105	16.0	1024	105
22	31.6	256	256	50.0	256	256	12.8	1025	106	19.6	1025	106
-	400%	400%	400%	608%	400%	400%	164%	1600%	164%	248%	800%	164%

TABLE 3.2 – Résultats de la méthode de résolution de la relaxation linéaire

Le tableau 3.2 nous indique que sur les différentes stratégies de génération FM est la meilleure.

On observe une variation du temps de calcul en fonction de la direction de la recherche du plus court chemin. En effet, la méthode B est 1.5 fois plus lente que la méthode F, car B génère 1.5 fois plus de labels pour trouver un chemin améliorant, ceci s'explique par la topologie de la grille pour qui une recherche partant de S aura après le premier voisinage le nombre de clients comme chemins qui se concurrenceront plus que les 2 chemins partant de OLT .

L'ajout des chemins non-dominés calculés lors de la recherche du plus court chemin a un impact significatif pour le temps d'exécution de l'algorithme. Ces méthodes sont 2.44 fois plus rapides que celles dont elles sont issues en raison d'un plus grand nombre de chemins générés à chaque itération qui conduit à un plus petit nombre d'itérations. Les méthodes recyclant les chemins génèrent 4 fois plus de chemins qui malgré le fait qu'ils ne soient pas tous utilisés contiennent des chemins améliorant appartenant à une solution optimale. Le facteur 4 est une moyenne, nous pensons que l'évolution de la taille de la grille n'est pas assez significative pour voir ce facteur évoluer. On constate que la méthode BM est 1.61 fois plus rapide que F ce qui s'explique par le fait que le recyclage accélère la convergence vers une solution optimale de la génération de colonnes de 2.44 fois et la recherche de OLT vers S ralentit la recherche de plus court chemin de 1.5.

Nous observons que la progression est linéaire en fonction du nombre de sites clients. Ceci peut s'expliquer par le fait que chaque site doit être relié par au moins un chemin à OLT .

Le point de difficulté de la résolution du problème $FTTH$ est de trouver un chemin améliorant respectant les contraintes techniques, son amélioration impacterait grandement le temps de calcul de la solution optimale du problème $FTTH$.

La procédure d'arrondis quand à elle offre des solutions dont le saut d'intégrité moyen est de 18% qui correspond au coût de la solution :

$$\frac{val(FTTH_{int}) - OPT(FTTH)}{OPT(FTTH)}$$

où $val(FTTH_{int})$ est le coût de la solution renvoyée par l'algorithme 6.

3.6 Conclusions

Les réseaux accès FTTH avec une architecture GPON implique des contraintes techniques qui rendent complexe l'optimisation de leurs déploiements. Dans ce chapitre, nous avons modélisé et développé un algorithme pour la résolution du problème de déploiement d'équipements réseaux optiques.

Nous parvenons à offrir une modélisation de ce problème qui prend en compte à la fois le problème de localisation des coupleurs optiques, celui du déploiement des câbles de fibres optiques et le routage des demandes des clients dans ce réseau. De plus nous proposons de respecter l'architecture GPON en nous imposant une qualité de service par le biais de contraintes techniques sur les chemins satisfaisant les demandes clients.

Nous mettons en œuvre des techniques de résolutions pour atteindre plus rapidement une bonne solution. Nous réalisons des simulations pour qualifier l'efficacité de ces techniques. Faire en sorte que ces modèles correspondent exactement à la situation opérationnelle n'est pas chose facile. Mais pour que ces modèles soient pertinents pour l'évaluation des performances nous avons pris la décision de générer des grilles, cette méthode a déjà été appliquée pour les problèmes de déploiement de réseaux de fibres optiques. Nous obtenons des solutions de bonne qualité en temps raisonnable. Les simulations effectuées nous permettent de souligner l'importance de la prise en charge des informations calculées lors de la résolution de l'algorithme de programmation linéaire pour améliorer la convergence de la procédure de génération de colonnes.

Notre modélisation reste flexible et permet d'apporter de nouveaux paramètres tels que l'ajout d'amplificateurs optiques comme équipements réseaux, l'étude de ces extensions de notre modélisation qui impliquent aussi de nouvelles règles d'ingénieries est une perspective pour ce problème. D'un autre côté, l'amélioration de la recherche du plus court chemin généralisé multi-contraint reste un axe de recherche. L'implémentation des algorithmes proposés dans [133] permettrait d'accélérer la résolution par programmation dynamique en pratique. L'utilisation d'algorithmes d'approximation pour la recherche de chemin est aussi une voie à explorer. Enfin, tester l'algorithme que nous proposons sur des instances réelles nous permettrait d'évaluer la qualité de solution par rapport aux méthodes déjà mises en place chez les opérateurs.

Problème de délai dans la conception de réseaux

4

Résumé

Le problème de flot avec contrainte de délai est NP-complet dans le cas du délai convexe [23] mais aussi dans le cas du délai linéaire [39]. Nous nous intéressons donc au cas proportionnel et proposons deux preuves de NP-complétude en nous basant sur celle de [23]. Nous présentons ensuite un algorithme d'approximation facteur 2 sur le graphe chemin avec demandes uniformes. Finalement nous présentons une heuristique par recherche locale, qui consiste à construire un ensemble de chemins supportant du flot que l'on améliore en détectant des chemins à supprimer ou à ajouter. Cette heuristique a été évaluée de façon expérimentale.

4.1 Contexte du problème de délais

Fournir une garantie de qualité de service (QoS) pour les délais d'acheminement des flux Internet sur l'ensemble d'un réseau est une tâche difficile. De nos jours, différents services transitent sur le réseau dont les principaux sont : la voix sur IP, les vidéo en streaming et les données du trafic. Chacun de ces flux possède ses propres exigences en terme de performance. Le terme QoS manque effectivement d'une définition précise et unifiée. L'*ETSI*¹ et l'*ITU*² définissent la QoS comme "la qualité perçue par l'utilisateur final" [1]. Le groupe de travail *IETF*³ donne une définition plus concrète de la qualité de service et la définit comme l'ensemble des services requis pour transporter convenablement un flux de trafic sur le réseau [41].

1. *European Telecommunications Standards Institute*

2. *Internationale Telecommunications Union*

3. *Internet Engineering Task Force*

En pratique le délai est l'indicateur le plus souvent utilisé pour mesurer la QoS.

Le délai mesure le temps passé par un paquet dans un réseau de télécommunication entre le moment où il est émis par le sommet source et le moment où il est reçu par le sommet destination. Il dépend de différents éléments dont chacun peut engendrer du retard. On identifie principalement quatre éléments pouvant retarder l'acheminement des messages : le délai d'insertion, le délai d'attente, le délai de traitement et le temps de propagation. Le délai d'insertion correspond au temps nécessaire pour insérer un paquet de taille donnée sur une installation de transmission. Le délai d'attente du tampon de sortie représente le temps d'attente avant d'être servi par le dispositif de transmission, ou plus concrètement le temps nécessaire au routeur pour examiner et acheminer le paquet et éventuellement effectuer d'autres opérations, telles que le cryptage ou la compression des données. La propagation correspond au temps nécessaire pour traverser le dispositif de transmission en fonction de la longueur de la liaison et du matériau, et de la longueur d'onde du signal (qu'il soit électrique ou électromagnétique). Étant donnée l'évolution matériel, les vitesses d'insertion et de transmission des paquets ont tellement progressé que la propagation et le délai d'attente (pour le calcul du routage) deviennent les facteurs limitant.

Le degré de congestion sur un lien est un élément qui affecte grandement le comportement des utilisateurs en perturbant le temps d'attente de leurs applications. Les mesures terrain [148] effectuées sur une base liaison $T3^4$ du réseau cœur appartenant à une entreprise de télécommunications basée au Royaume-Uni sont représentées sur l'image 4.1.

Ce graphique représente la moyenne du délai sur plusieurs jours d'un aller-retour de paquet en fonction de la charge du lien. La courbe qui forme les valeurs moyennes peut être très bien approximée par la fonction convexe introduite par Kleinrock dont la justification théorique provient de la théorie des files d'attente [106]. Cette fonction définit le délai si l'arrivée des paquets suit une loi de Poisson et que la taille de la mémoire tampon est infinie. Soit c_e la capacité de transfert du lien e et x_e son utilisation, on définit la fonction de délai $f : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, $x_e \mapsto \frac{1}{c_e - x_e}$.

4. T -carrier de niveau 3 soit environs 45 Mbit/s

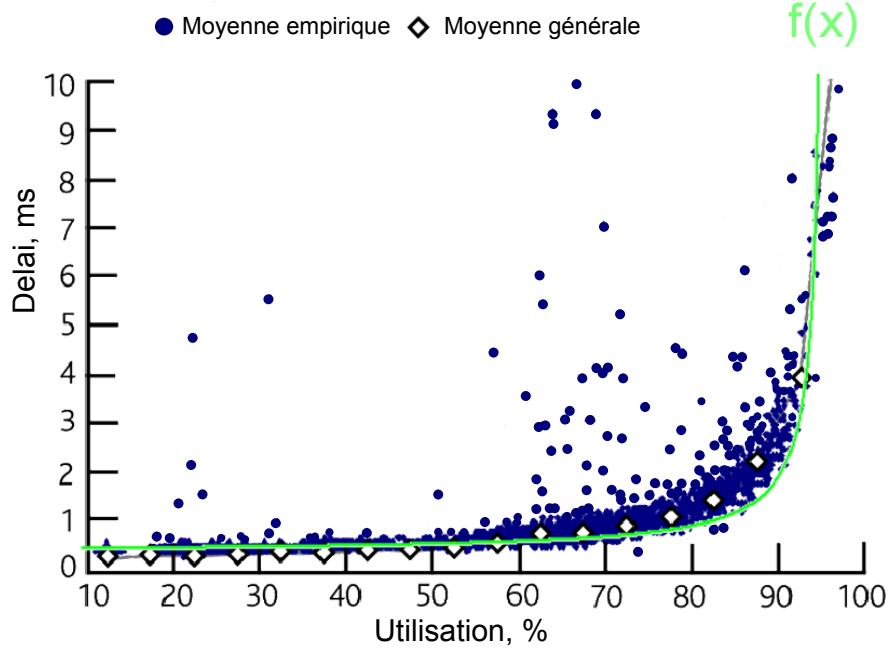


FIGURE 4.1 – Approximation du délai par la fonction : $f(x_e) = \frac{1}{c_e - x_e}$

État de l'art

La métrique prépondérante dans l'étude de *QoS* dans le domaine de l'optimisation de la conception de réseaux de télécommunications est le délai. Une étude approfondie sur les modélisations de ce type de problème est proposée dans les documents [107, 140].

Pour la réduction du délai moyen sur le réseau, dans le cas où le trafic entre une paire origine-destination peut être acheminé sur plusieurs chemins, les algorithmes de Fratta et al. [66] et de LeBlanc [115] permettent de résoudre de manière heuristique ce problème. Il s'agit d'un cas particulier de la méthode de Frank-Wolfe pour résoudre les problèmes d'optimisation non-linéaire ayant des contraintes linéaires.

Si la modélisation du délai est une fonction convexe, Ouorou et al. [128, 129] décrivent un algorithme d'approximation basé sur le retard moyen. En se basant sur le fait que la fonction convexe est positive Galtier et al. [67] proposent une modélisation du problème sous la forme d'un programme semi-défini positif et un algorithme de génération de colonnes pour le résoudre.

De nombreuses variantes de ce problème ont suivi telles que : la prise en compte

du coût des capacités avec [70, 134, 101] et en ajoutant la modification de la topologie du réseau [19, 23].

Plusieurs auteurs ont considéré le délai de bout en bout le long du chemin reliant l'origine du paquet à sa destination.

Dans le document [153], les auteurs considèrent un réseau avec topologie donnée ainsi que les capacités en se basant sur les travaux de [111]. Le problème est de choisir un unique chemin pour chaque paire origine-destination, de telle sorte que les contraintes de capacité et de délai soient satisfaites. L'objectif étant de minimiser le délai moyen global dans le réseau. Leur solution utilise la relaxation lagrangienne pour prendre en compte les contraintes dans la fonction objectif et décompose le problème en un sous-problème de plus court chemin pour chaque paire origine-destination et un sous-problème non-linéaire pour chaque liaison. Une optimisation par sous-gradient est appliquée et une heuristique est utilisée pour générer une solution primale réalisable. Les instances utilisées pour les tests expérimentaux disposent d'environ 30 nœuds et les demandes sont de l'ordre de 1500.

Les mêmes auteurs [152] examinent le problème de dimensionnement et de routage dans le backbone. La capacité de chaque liaison doit être choisie parmi un ensemble discret et il est nécessaire d'acheminer le trafic sur un chemin unique pour chaque paire origine-destination. Les auteurs formulent ce problème en un programme mathématique non-convexe où pour chaque lien une variable binaire indique si la liaison est inclus dans la solution. En plus d'une contrainte de délai moyen, ils ajoutent une contrainte de retard de bout en bout pour chaque paire origine-destination. Des contraintes supplémentaires assurent que le réseau est connexe et que les commutateurs au niveau des nœuds sont dimensionnés pour gérer le nombre de liens entrants et le trafic. L'objectif étant de minimiser le coût total de l'installation du réseau, y compris le coût de la capacité de liaison et le coût de la capacité de commutateurs. Une solution basée sur la relaxation lagrangienne est proposée, au cours de laquelle cinq sous-problèmes indépendants doivent être résolus au niveau de chaque itération de sous-gradient. Les tests expérimentaux présentés utilisent des instances d'au maximum 30 nœuds et 10 demandes.

Par la suite une méthode de Branch-and-Price pour le délai avec une contrainte de k -divisibilité de routage a été proposée dans [146]. Le problème de k -divisibilité, introduit pour la première fois par [11], consiste à utiliser au plus k chemins pour chaque paire source-destination. Dans ce travail, une contrainte de délai est ajoutée sur les chemins générés dynamiquement. Les tests expérimentaux sont effectués sur des instances relativement petites d'au maximum 15 nœuds avec une seule demande.

Récemment, [85] propose un modèle non-linéaire en nombres entiers. Dans un premier temps les auteurs décrivent l'enveloppe convexe de programme non-linéaire en nombre entier dont la résolution se fait via une approximation de ce modèle par un modèle de programme linéaire en nombres entiers. Les tests expérimentaux présentés utilisent des instances de graphes possédant une centaine de sommets, les demandes sont de l'ordre de 500 mais pour chaque demande l'ensemble des chemins pouvant supporter le flots est fixé (10 au maximum par demandes).

4.2 Problème de flot avec contrainte de délai convexe

Dans cette section, nous commençons par donner une définition formelle du problème de flot avec contrainte de délai convexe. Nous donnons une modélisation de ce problème en un programme mathématique non-linéaire. Nous montrons qu'il est possible d'effectuer une transformation classique des contraintes non-linéaires pour obtenir un programme linéaire. Une solution optimale du programme linéaire sera une approximation de la solution optimale du problème de flot avec contrainte de délai convexe.

Soient un digraphe $G = (V, E)$ et $\mathcal{K}$ l'ensemble des demandes. À toute demande $k \in \mathcal{K}$ nous associons un couple $\{s_k, t_k\} \subseteq V$ où s_k représente la source et t_k le puits de la demande k ; et d_k représente la valeur de la demande. Soit $\mathcal{P}$ l'ensemble des chemins entre les sources et les destinations pouvant supporter du flot. Nous notons ϕ_p la quantité non négative de flot qui circule sur le chemin $p \in \mathcal{P}$. Nous notons $k(p)$ la demande $k \in \mathcal{K}$ associé au chemin $p \in \mathcal{P}$. À chaque arc $e \in E$ est assigné : une capacité C_e qui représente la quantité maximum de flot que peut supporter l'arc e ; et une fonction de délai convexe croissante de l'arc $F_e : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ qui permet de calculer le délai $F_e \left(\sum_{p:e \in p} \phi_p \right)$ de l'arc e en fonction de la quantité de flot $\sum_{p:e \in p} \phi_p$ qui y circule.

Nous ajoutons une contrainte de délai sur les chemins qui portent une quantité positive de flot. Si nous décidons d'envoyer sur un chemin $p \in \mathcal{P}$ une quantité non nulle de flot nous imposons que le temps total de circulation du flot sur p soit inférieur à une borne donnée. On peut supposer que cette borne est unitaire moyennant une mise à l'échelle des fonctions F_e . Au contraire, aucune contrainte n'est imposée sur le chemin p si aucun flot ne circule sur ce chemin.

CHAPITRE 4. PROBLÈME DE DÉLAI DANS LA CONCEPTION DE RÉSEAUX

Le problème du *flot avec contrainte de délai convexe* peut être formulé de la façon suivante :

Problème:	Flot à contrainte de délai convexe
Entrée:	Un digraphe $G = (V, E)$, l'ensemble des demandes $\mathcal{K}$, pour chaque arc une capacité et une fonction de délai convexe croissante
Question:	S'il existe, trouver un flot dans G satisfaisant les demandes $\mathcal{K}$ et les chemins supportant le flot respectent la contrainte de délai ?

Nous proposons une formulation en programme mathématique non-linéaire FDC du problème de flot avec contrainte de délai convexe.

$$FDC = \left\{ \begin{array}{ll} \text{Max} & \sum_{p \in \mathcal{P}} \phi_p \\ \text{s.c} & \\ & \sum_{e \in q} F_e \left(\sum_{p: e \in p} \phi_p \right) \leq 1, \quad \forall q \in \mathcal{P} \text{ si } \phi_q > 0 \quad (1) \\ & \sum_{p: e \in p} \phi_p \leq C_e, \quad \forall e \in E \quad (2) \\ & \sum_{p: k(p)=k} \phi_p \leq d_k, \quad \forall k \in \mathcal{K} \quad (3) \\ & \phi_p \geq 0, \quad \forall p \in \mathcal{P} \end{array} \right.$$

La fonction objectif consiste à maximiser la somme totale du flot porté par les chemins. La contrainte (1) assure que les chemins supportant un flot non nul ne dépasse pas le délai maximum autorisé. Les contraintes (2) obligent le flot sur chaque liaison e à respecter la capacité de cette liaison. Les contraintes (3) prennent en compte les demandes $\mathcal{K}$ en bornant à d_k le flot des chemins satisfaisant la demande $k \in \mathcal{K}$. Une instance du problème de flot à délai convexe a une réponse oui si et seulement si le programme linéaire FDC correspondant a une solution réalisable telle que $\sum_{p \in \mathcal{P}} \phi_p = \sum_{k \in \mathcal{K}} d_k$.

ε -approximation en programme linéaire

Le programme mathématique FDC est constitué d'une fonction objectif linéaire, d'une contrainte non-linéaire (1) et de deux contraintes linéaires (2) et (3). Dans cette section nous décrivons une transformation classique permettant d'approximer la contrainte non-linéaire (1) par un ensemble de contraintes linéaires.

Nous affirmons la proposition suivante.

Proposition 4.2.1. *Nous pouvons obtenir une ε -approximation de la fonction de délai convexe croissante de FDC par un ensemble de fonctions linéaires en temps polynomial dans la taille de l'instance et de n le plus petit entier tel que $F_e(\varepsilon n) \geq 1$ pour tout arc $e \in E$.*

Démonstration 4.2.1.

Soient $\varepsilon > 0$ et n tel que $F_e(\varepsilon n) \geq 1$ pour tout arc $e \in E$. Nous considérons pour chaque arc e , n fonctions linéaires $f_{e,i} : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ pour $i = \{1, \dots, n\}$ telles que $f_{e,i}(i-1) = F_e(\varepsilon(i-1))$ et $f_{e,i}(i) = F_e(\varepsilon i)$. Soit α_i le coefficient directeur de la fonction $f_{e,i}$. L'équation de la fonction $f_{e,i}(x) = \sum_{j=1}^{i-1} (\alpha_j - \alpha_{j+1})(\varepsilon j) + \alpha_i x$, $x \in \mathbb{R}^+$.

Nous obtenons le programme linéaire FDL à partir de FDC de la manière suivante. Pour chaque arc e nous associons une nouvelle variable $z_e \geq 0$ et remplaçons les contraintes (1) par les contraintes suivantes :

$$\begin{cases} \sum_{e \in p} z_e \leq 1, & \forall p \in \mathcal{P} \text{ si } \phi_p > 0 \quad (1.1) \\ \sum_{j=1}^{i-1} (\alpha_j - \alpha_{j+1})(\varepsilon j) + \alpha_i \sum_{p: e \in p} \phi_p \leq z_e & \forall e \in E, \forall i \in 1, \dots, n \quad (1.2) \end{cases}$$

Toute solution de FDC est réalisable dans FDL . En effet, pour chaque arc e , $\forall i, j \in \{1, \dots, n\}$ avec $i \neq j$ nous avons $f_{e,i}(x') \geq f_{e,j}(x')$, pour $x' \in [(i-1)\varepsilon, i\varepsilon]$. Pour chaque solution de FDL , pour chaque arc $e \in E$ il y a au plus ε en excès. Nous pouvons donc construire une solution réalisable pour FDC en supprimant au plus ε unités de flot pour chaque arc. Donc nous avons bien une ε -approximation de la fonction de délai convexe croissante pour chaque arc de E .

Le nombre de contraintes en plus dans FDL est de $|E|n$. Or n est linéairement dépendant de ε . Cette transformation est donc polynomial dans la taille de l'instance et de ε .

□

La figure 4.2 représente la construction précédente. L'abscisse représente le flot sur un arc e et l'ordonnée représente le délai de e . Nous effectuons un ε -approximation de la fonction convexe de délai pour $\varepsilon = 1$. On remarque que la fonction linéaire $f_{e,8}(x)$ intersecte la fonction convexe pour $x = 7$ et $x = 8$. De plus $f_{e,8}(x') \geq f_{e,j}(x')$ pour $1 \leq j \leq 9$ et $x' \in [7, 8]$.

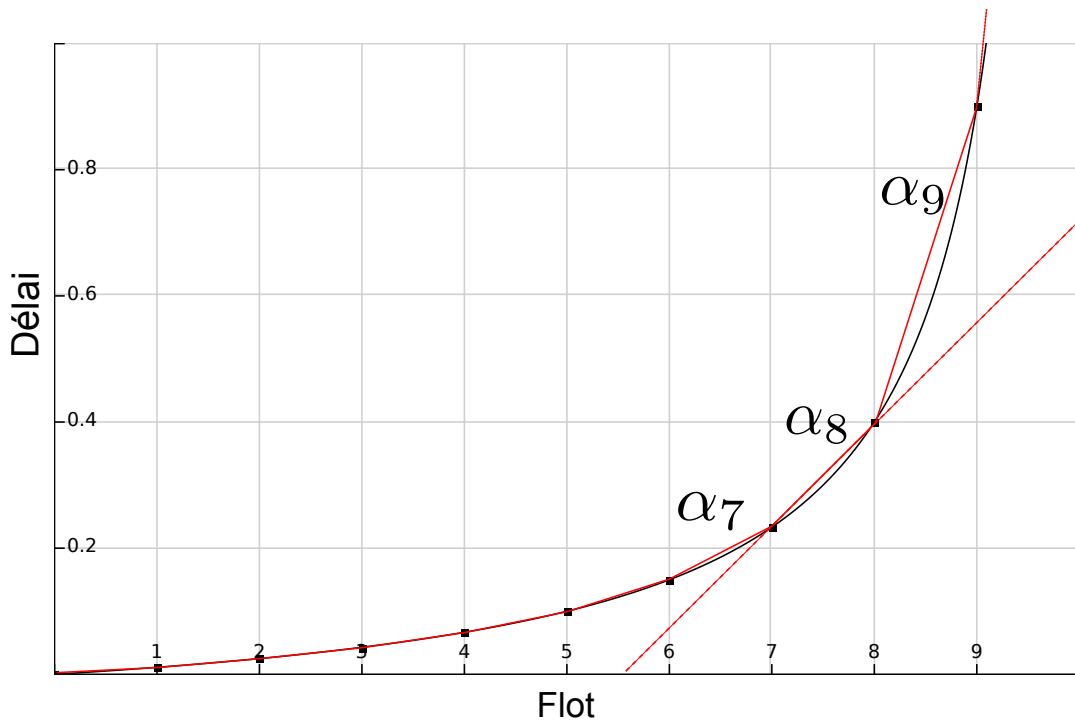


FIGURE 4.2 – ε -approximation d'une contrainte convexe par des contraintes linéaires

4.3 Problème de flot avec contrainte de délai proportionnel

À partir de cette section et jusqu'à la fin du chapitre, nous nous intéressons au cas où le délai est proportionnel à la quantité de flot qui circule sur cet arc.

Nous notons, pour chaque arc e , α_e le coefficient de proportionnalité de l'arc e . Si $\sum_{p:e \in p} \phi_p$ unités de flot circulent sur l'arc e , alors le délai de l'arc e sera de $\alpha_e \sum_{p:e \in p} \phi_p$. La difficulté de ce problème réside dans le choix des chemins sur lesquels un flot non-nul circule. Ce problème est NP-difficile et ce même si on supprime la contrainte de capacité ou la contrainte de demande. Nous notons FDP le problème de flot avec délai proportionnel qui correspond au programme mathématique FDC dont la contrainte (1) est remplacé par la contrainte suivante :

$$\left\{ \alpha_e \sum_{p:e \in p} \phi_p \leq 1, \quad \forall p \in \mathcal{P} \text{ si } \phi_p > 0 \right. \quad (1.3)$$

Nous présentons deux démonstrations de NP-complétudes obtenues en adaptant au cas linéaire la réduction utilisée dans [23] pour le cas avec des fonctions proportionnelles. Dans les deux cas la réduction se fait à partir du problème Partition.

Problème de multiflot sans capacité et avec contrainte de délai proportionnel

Nous considérons le problème de multiflot sans capacité et avec contrainte de délai proportionnel ($FDP-2$) correspondant au programme mathématique FDP sans la contrainte (2) de capacité. Le flot sera, ici, contraint par le délai au lieu d'être contraint par les capacités des arcs.

Proposition 4.3.1. *Le problème de multiflot sans capacité et avec contrainte de délai proportionnel est NP-Difficile.*

Démonstration 4.3.1.

Nous effectuons pour ce problème une réduction polynomiale du problème Partition. Le problème Partition étant NP-Complet, notre problème est NP-difficile. Cette réduction est basée sur la construction suivante.

Soit $X = \{s_1, \dots, s_n\}$ une instance du problème Partition et $\mathcal{S} = \sum_{s_i \in X} s_i$. Nous définissons le digraphe G de la manière suivante. Les sommets $V(G) := (\{a, b, t, z, \} \cup \{v_1, v_2, \dots, v_n\})$. L'ensemble des arcs inclut :

- les arcs (a, t) , (b, t) avec les coefficients $\alpha_{(a,t)} = \alpha_{(b,t)} = \frac{2}{\mathcal{S}}$;

- les arcs (a, z) , (b, z) avec les coefficients $\alpha_{(a,z)} = \alpha_{(b,z)} = 0$;
- pour tout $i \in \{1, \dots, n\}$ les arcs (v_i, a) , (v_i, b) de coefficients $\alpha_{(v_i,a)} = \alpha_{(v_i,b)} = \frac{1}{s_i}$.

La figure 4.3 représente la construction précédente.

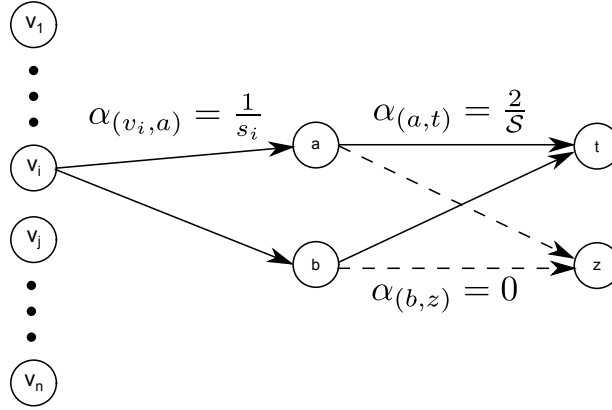


FIGURE 4.3 – Graphe d’une instance du problème Partition pour le problème de multiflot sans capacité avec contrainte de délai proportionnel

Nous définissons dans le graphe G une instance I du problème de multiflot avec délai borné dont le délai maximum est 2 et les demandes $\mathcal{K}$ sont :

- de valeurs s_i entre v_i et t pour $v_i \in \{v_1, \dots, v_n\}$
- de valeurs $2s_i$ entre v_i et z pour $v_i \in \{v_1, \dots, v_n\}$

Pour commencer montrons qu’une solution de l’instance du problème de multiflot I permet de construire une solution de l’instance X du problème de Partition. Nous définissons $\phi \in \mathbb{R}^E$ le flot d’une solution de I . Pour simplifier les calculs nous définissons aussi $d_e = \phi_e \alpha_e$ comme le délai de l’arc e .

Premièrement nous montrons que si le chemin $\{v_i, a, t\}$ supporte du flot alors le chemin $\{v_i, b, t\}$ n’en supporte pas. Par définition de l’instance I , $\phi_{(a,t)} + \phi_{(b,t)} = \mathcal{S}$. Donc, l’une des deux inégalités suivantes est vérifiée : $\phi_{(a,t)} \geq \frac{\mathcal{S}}{2}$, ou $\phi_{(b,t)} \geq \frac{\mathcal{S}}{2}$.

Supposons sans perte de généralité que $\phi_{(a,t)} \geq \frac{\mathcal{S}}{2}$. Par définition le délai $d_{(a,t)}$ est égale à $\phi_{(a,t)} \alpha_{(a,t)}$, nous obtenons donc $d_{(a,t)} \geq 1$. Le chemin $\{v_i, a, t\}$ supporte du flot, par conséquent nous avons $d_{(v_i,a)} + d_{(a,t)} \leq 2$, et $d_{(v_i,a)} \leq 1$, ce qui implique $\phi_{(v_i,a)} \alpha_{(v_i,a)} \leq 1$, donc $\phi_{(v_i,a)} \leq s_i$. Or par construction la somme des demandes dont l’origine est v_i est égale à $3s_i$, par conséquent $\phi_{(v_i,b)} \geq 2s_i$, donc $d_{(v_i,b)} \geq 2$. La contrainte de délai maximum étant de 2, l’arc (b, t) ayant un coefficient non nul, le chemin $\{v_i, b, t\}$ ne peut pas supporter de flot. Nous allons maintenant montrer

4.3. PROBLÈME DE FLOT AVEC CONTRAINTES DE DÉLAI PROPORTIONNEL

qu'il est facile de trouver une partition des sommets $\{v_1, \dots, v_n\}$ correspondant à l'utilisation de a ou de b pour satisfaire les demandes de t . De plus, nous montrons que la somme des unités de flot routées par a est égale à la somme de celles routées par b . Nous venons de montrer que si un chemin $\{v_i, a, t\}$ a un flot non nul alors $\{v_i, b, t\}$ ne supporte pas de flot, la réciproque est vraie aussi (en utilisant la même démonstration). Par conséquent chaque v_i "choisit" un chemin passant soit par a , soit par b et route la totalité des s_i unités de la demande entre v_i et t via ce sommet. Nous remarquons que le graphe support du flot satisfaisant les demandes des v_i vers t , privé de t , forme deux composantes connexes $\mathcal{C}_a$ et $\mathcal{C}_b$ contenant respectivement le sommet a ou b . De plus, pour chaque v_i selon le chemin "choisi" l'une des deux inégalités suivantes est vérifiée : $d_{(v_i, a)} = 1$, ou $d_{(v_i, b)} = 1$. Or $d_{(v_i, a)} + d_{(a, t)} \leq 2$ et $d_{(v_i, b)} + d_{(b, t)} \leq 2$. Alors $d_{(a, t)} \leq 1$ et $d_{(b, t)} \leq 1$, ce qui implique $\phi_{(a, t)} \leq \frac{\mathcal{S}}{2}$ et $\phi_{(b, t)} \leq \frac{\mathcal{S}}{2}$. Or $\phi_{(a, t)} + \phi_{(b, t)} = \mathcal{S}$, donc $\phi_{(a, t)} = \phi_{(b, t)} = \frac{\mathcal{S}}{2}$. Nous avons donc montré que pour une solution de I le flot satisfaisant la demande de t est de valeur $\mathcal{S}$ dont l'ensemble des chemins $\{v_i, \dots, t\}$ qui supportent respectivement s_i unités de flot se répartissent à part égale sur les arcs (a, t) et (b, t) . Ce qui implique que la valeur du flot sur chacun de ces arcs est $\frac{\mathcal{S}}{2}$ et que l'instance du problème Partition a une réponse oui. Réciproquement, nous allons montrer que si l'instance X du problème Partition a une réponse oui, alors I a une réponse oui. Toute solution de X est une solution de I . Soit une partition (A, B) valide du problème Partition, la construction de la solution de I se fait de la manière suivante : nous routons s_i unités de flot de v_i vers t via a si $i \in A$ ou via b si $i \in B$ et les $2s_j$ unités de flot de v_i vers z via a si $i \in B$ ou via b si $i \in A$. Nous obtenons un flot qui respecte les contraintes de délai.

Clairement cette transformation est polynomiale. Le problème Partition se réduit donc polynomialement au problème de flot avec délai proportionnel.

□

Problème de flot avec capacité et contrainte de délai linéaire

Nous considérons le problème de de flot avec contrainte de délai proportionnel ($FDP-3$) correspondant au programme mathématique FDP sans la contrainte (3) de demande.

Proposition 4.3.2. *Le problème de flot avec capacité et contrainte de délai linéaire est NP-Difficile.*

Comme pour le problème précédent, il est possible d'effectuer une réduction polynomiale du problème Partition en notre problème. Cette démonstration similaire à la précédente se base sur la construction de graphe suivante.

Soit $X = \{s_1, \dots, s_n\}$ une instance du problème Partition. Nous définissons le graphe G' de la manière suivante. Les sommets $V(G') := (\{S, a, b, t, z, P\} \cup \{v_1, v_2, \dots, v_n\})$. L'ensemble des arcs inclut :

- les arcs (a, t) , (b, t) de coefficient $\alpha_{(a,t)} = \alpha_{(b,t)} = \frac{2}{\mathcal{S}}$
et de capacité $c_{(a,t)} = c_{(b,t)} = \infty$;
- les arcs (a, z) , (b, z) de coefficient $\alpha_{(a,z)} = \alpha_{(b,z)} = 0$
et de capacité $c_{(a,z)} = c_{(b,z)} = \infty$;
- les arcs (t, P) , (z, P) de coefficient $\alpha_{(t,P)} = \alpha_{(z,P)} = 0$
et de capacité $c_{(t,P)} = \mathcal{S}$, $c_{(z,P)} = 2\mathcal{S}$;
- pour tout $i \in \{1, \dots, n\}$,
 - l'arc (v_i, a) de coefficient $\alpha_{(v_i,a)} = \frac{1}{s_i}$ et de capacité $c_{(v_i,a)} = \infty$;
 - l'arc (v_i, b) de coefficient $\alpha_{(v_i,b)} = \frac{1}{s_i}$ et de capacité $c_{(v_i,b)} = \infty$;
 - l'arc (S, v_i) de coefficient $\alpha_{(S,v_i)} = 0$ et de capacité $c_{(S,v_i)} = 3s_i$.

La figure 4.4 représente la construction précédente.

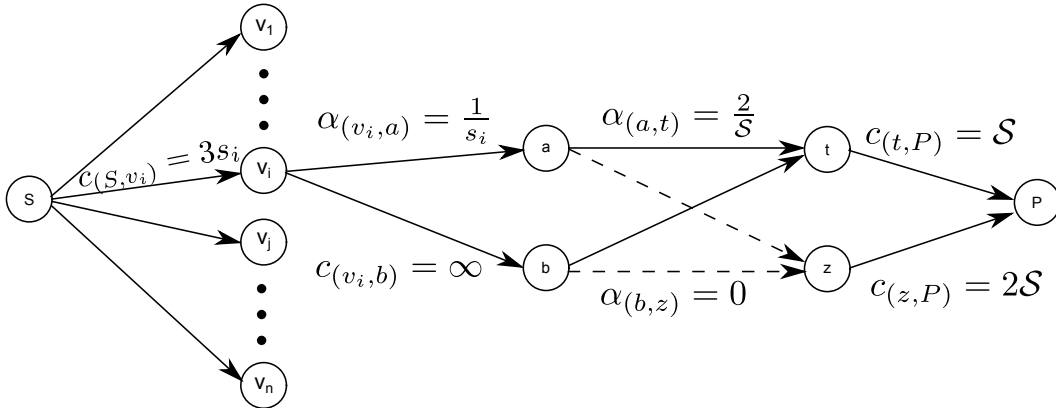


FIGURE 4.4 – Graphe d'une instance du problème Partition pour le problème de flot avec capacité et contrainte de délai proportionnel

Nous constatons que le graphe ainsi construit pour la démonstration est similaire au précédent. En effet nous avons utilisé la transformation de base d'un problème de flot multi-sources multi-puits en un flot simple sous la condition que les sommets origines des demandes sont disjoints des sommets destinations. Nous avons déjà

utilisé cette transformation dans le Chapitre 3. Ceci implique immédiatement la preuve de cette réduction. De plus il est trivial pour une même instance du problème Partition X de construire le graphe G correspondant à la réduction du problème précédent et le graphe G' correspondant à la réduction décrite ci-dessus. Si l'instance X a la réponse oui et ϕ la solution optimale du problème précédent pour le graphe G , nous pouvons construire une solution optimale pour notre problème dans le graphe G' . Pour cela nous affectons la valeur ϕ_e pour $e \in E(G') \cap E(G)$ et saturons les autres arcs $e_2 \in E(G') - E(G)$. Nous avons donc une solution maximale qui sature tout les arcs de G , qui respecte la conservation du flot et qui respecte la contrainte de délai.

4.4 Algorithme d'approximation facteur 2 pour le graphe chemin

Dans cette section nous présentons un algorithme d'approximation de facteur 2 pour le problème de flot avec délai proportionnel sans capacité FDP-2 dans le cas où le graphe en entrée est un chemin.

Modélisation du problème

Soit $\mathcal{P}_n$ le chemin sur n sommets, nous notons $V(\mathcal{P}_n) = \{v_1, \dots, v_n\}$ l'ensemble de ses sommets et $E(\mathcal{P}_n)$ l'ensemble de ses arêtes $e_i = v_i v_{i+1}$ pour $i = 1, \dots, n-1$. Nous considérons une famille de flots entre m paires source-destination $s_k, t_k \in V(\mathcal{P}_n)$, $k = 1, 2, \dots, m$. Nous notons ϕ_k la quantité (non négative) de flot qui circule sur l'unique chemin P_k entre s_k et t_k dans $\mathcal{P}_n$. Nous supposons que le temps τ_e pour traverser un arc $e \in E(\mathcal{P}_n)$ est proportionnel à la quantité totale de flot $\sum_{k:e \in E(P_k)} \phi_k$

qui circule sur cet arc i.e. $\tau_e := \alpha_e \sum_{k:e \in E(P_k)} \phi_k$. Le facteur α_e pour chaque arc e est une donnée du problème. Si nous décidons d'envoyer entre s_k et t_k une quantité non nulle de flot, nous imposons que le temps total de circulation du flot entre s_k et t_k soit inférieur à une borne donnée (nous supposons cette borne est unitaire). Cela implique que pour tout $k = 1, \dots, m$ tel que $\phi_k > 0$, la contrainte suivante doit être vérifiée :

$$\sum_{e \in E(P_k)} \left(\alpha_e \sum_{k:e \in E(P_k)} \phi_k \right) \leq 1.$$

Au contraire, si aucun flot ne circule entre s_k et t_k la contrainte précédente peut être violée.

Le problème du *flot à délai proportionnel* consiste à trouver parmi toutes les solutions $(\phi_1, \phi_2, \dots, \phi_m)$ qui vérifient cette condition, une solution pour laquelle le flot total $\sum_{i=1}^m \phi_i$ est maximal. La difficulté de ce problème réside dans le choix des chemins sur lequel un flot non-nul circule. En effet, une fois ce choix effectué les valeurs ϕ_k non nulles peuvent être obtenues en résolvant le programme linéaire (P) suivant :

$$(P) \left\{ \begin{array}{ll} \text{Max} & \sum_k \phi_k \\ \text{s.c} & \\ & \sum_{e \in P_i} \alpha_e \left(\sum_{k: e \in P_k} \phi_k \right) \leq 1 \quad \forall i = 1, \dots, m \\ & \phi_k \geq 0 \quad \forall k = 1, \dots, m. \end{array} \right.$$

Le dual (D) de ce programme linéaire s'écrit :

$$(D) \left\{ \begin{array}{ll} \text{Min} & \sum_i y_i \\ \text{s.c} & \\ & \sum_{e \in P_k} \alpha_e \left(\sum_{i: e \in P_i} y_i \right) \geq 1 \quad \forall k = 1, \dots, m \\ & y_i \geq 0 \quad \forall i = 1, \dots, m. \end{array} \right.$$

Nous allons maintenant décrire un algorithme qui permet d'obtenir en temps polynomial une solution du problème de flot à délai proportionnel dont la valeur est au minimum la moitié de celle d'une solution optimale. Pour cela nous allons considérer des solutions ϕ dites *sans intersection* dont le support $\{P_k : \phi_k > 0\}$ est constitué de chemins deux à deux disjoints. L'algorithme que nous allons présenter est basé sur le lemme suivant.

Lemme 4.4.1. *Pour toute instance I du problème de flot à délai proportionnel, le flot total de la meilleure solution sans intersection est au moins la moitié du flot total d'une solution optimale.*

Le problème de calculer la meilleure solution sans intersection est clairement polynomial car il se réduit au problème de calculer un stable de poids maximum dans un graphe d'intervalles [79]. Les sommets de G sont associés aux chemins $P_1, P_2, \dots, P_m$

4.4. ALGORITHME D'APPROXIMATION FACTEUR 2 POUR LE GRAPHE CHEMIN

et sont munis de poids w_k qui représentent la quantité maximale de flot qui peut circuler sur le chemin P_k si aucun arc de ce chemin n'est utilisé par un autre flot, i.e. $w_k := 1 / \sum_{e \in E(P_k)} \alpha_e$. Les arêtes de G sont les paires de chemins qui partagent au moins une arête.

Pour obtenir un algorithme d'approximation facteur 2 pour le problème de flot à délai proportionnel, il suffit donc de démontrer le Lemme 4.4.1.

Démonstration du lemme 4.4.1. Considérons une solution optimale ϕ^* de l'instance I et l'instance I' obtenue à partir de I en supprimant toutes les demandes qui ne sont pas dans le support de la solution optimale, i.e. toutes les demandes k telles que $\phi_k^* = 0$. Toute solution de l'instance I' étant une solution de I , ϕ_I^* est donc également une solution optimale pour I' . Si nous pouvons montrer que l'affirmation du Lemme 1 est vérifiée pour I' alors nous aurons montré qu'elle est vraie également pour I . En effet, la meilleure solution $\phi_{I'}$ sans intersection pour I' est une solution sans intersection pour I , si nous notons ϕ_I la meilleure solution sans intersection pour I , nous avons :

$$f(\phi_I) \geq f(\phi_{I'}) \geq \frac{1}{2} \cdot f(\phi_I^*).$$

La particularité de l'instance I' est qu'il existe une solution optimale de I' dans laquelle toutes les contraintes de délai sont actives. Cette solution est donc une solution optimale du programme linéaire (P) et sa valeur est donc au plus celle de n'importe quelle solution y du dual (D) de (P) . Nous allons montrer qu'il existe une solution sans intersection $\phi_{I'}$ de I' et une solution admissible $y_{I'}$ du programme linéaire dual (D) telle que la valeur de $y_{I'}$ soit au plus le double de celle de $\phi_{I'}$.

Soit $\mathcal{C}$ l'ensemble des cliques maximales du graphe G défini ci-dessus. Ce graphe étant parfait, nous pouvons calculer en temps polynomial un stable de poids maximal dans G . De plus, nous savons que le vecteur caractéristique z d'un tel stable est une solution optimale du programme linéaire (P') suivant :

$$(P') \left\{ \begin{array}{ll} \text{Max} & \sum_k z_k w_k \\ \text{s.c} & \\ & \sum_{k: P_k \in C} z_k \leq 1 \quad \forall C \in \mathcal{C} \\ & z_k \geq 0 \quad \forall k = 1, \dots, m' \end{array} \right.$$

Considérons le dual (D') de ce programme linéaire :

$$(D') \left\{ \begin{array}{l} \text{Min} \quad \sum_{C \in \mathcal{C}} t_C \\ s.c \quad \sum_{C: P_k \in C} t_C \geq w_k \quad \forall k = 1, \dots, m' \\ t_C \geq 0 \quad \forall C \in \mathcal{C} \end{array} \right.$$

Soit t une solution optimale du dual (D') . Nous allons montrer que l'algorithme suivant permet de calculer à partir de t , une solution y admissible pour le programme linéaire (D) dont le coût est au plus le double de celui de t . Pour $C \in \mathcal{C}$, nous notons $i(C)$ (resp. $j(C)$) l'indice du chemin le plus à gauche (resp. le plus à droite) dans la clique C .

1. Pour tout $i = 1, \dots, m$ faire
2. $y_i := 0$
3. Pour tout $C \in \mathcal{C}$ faire
4. $y_{i(C)} := y_{i(C)} + t_C$
5. $y_{j(C)} := y_{j(C)} + t_C$

Pour tout $k = 1, \dots, m$, si $e \in P_k$ alors les inégalités suivantes sont vérifiées :

$$\sum_{i: e \in P_i} y_i \geq \sum_{C: P_k \in C} t_C \geq w_k.$$

La deuxième inégalité est trivialement vérifiée car t est une solution de D' . Pour vérifier la première inégalité, nous commençons par remarquer que si le chemin P_k appartient à la clique C alors $e \in E(P_k)$ implique $e \in E(P_{i(C)})$ ou $e \in E(P_{j(C)})$. Par conséquent, si une clique C contribue t_C pour la deuxième somme alors $y_{i(C)}$ ou $y_{j(C)}$ contribue t_C pour la première somme. Nous en concluons que

$$\sum_{e \in P_k} \alpha_e \left(\sum_{i: e \in P_i} y_i \right) \geq \sum_{e \in P_k} \alpha_e w_k = w_k \sum_{e \in P_k} \alpha_e = 1.$$

Ce qui montre que y est une solution admissible de (D) . Clairement sa valeur est au plus deux fois le poids du stable de poids maximum. Soit S le stable maximum dont le vecteur caractéristique est une solution optimale de (P') . La solution sans intersection correspondante est donc une 2-approximation pour l'instance I' ce qui termine la preuve du Lemme 1.

□

4.5 Modélisation en un programme linéaire disjonctif

Nous proposons une heuristique de recherche locale basée sur une modélisation par programmation disjonctive du problème de flot avec contrainte de délai proportionnel décrit dans 4.3. La programmation disjonctive, introduite dans la section 9, est une manière simple de modéliser notre problème de flot avec contrainte de délai proportionnel, qui comme nous l'avons montré précédemment (cf. 4.3) à comme difficulté le choix de l'ensemble des chemins supportant le flot.

Le problème mathématique FDC n'étant pas défini formellement, car modélisé par rapport à une solution courante. Nous sommes amenés à définir un coefficient multiplicateur sur les contraintes qui définit si un chemin est actif ou non.

Soit une instance du problème de flot avec contrainte de délai proportionnel et capacité, nous définissons I un ensemble d'indices de taille $2^{|\mathcal{P}|}$ où $\mathcal{P}$ est la famille des chemins possibles de l'instance du problème. Soit $Z \in \{0, 1\}^{|\mathcal{P}| \times |I|}$ une matrice booléenne représentant toutes les combinaisons possibles de l'ensemble $\mathcal{P}$ tel que pour tout élément $i \in I$ et $p \in \{0, \dots, |\mathcal{P}|\}$ nous définissons $z_p^i \in Z$ tel que $z_p^i = 1$ si le chemin p est dans la combinaison i et $z_p^i = 0$ sinon.

L'élément z_p^i permet donc d'associer à une combinaison $i \in I$ et un indice de chemin p son état d'activité, dans le sens où un chemin "actif" pourra supporter du flot mais en contrepartie devra respecter la contrainte de délai maximum. Le programme sous la forme normale disjonctive suivant modélise le problème de flot avec contrainte de délai proportionnel et capacité (FDC).

$$DP = \left\{ \begin{array}{l} \text{Max} \quad \sum_{p \in \mathcal{P}} \phi_p \\ \text{s.c} \\ \bigvee_{i \in I} \left(\begin{array}{l} z_q^i \left(\sum_{e \in q} \alpha_e \sum_{p: e \in p} \phi_p z_p^i \right) \leq 1, \quad \forall q \in \mathcal{P} \quad (1) \\ \sum_{p: e \in p} \phi_p z_p^i \leq C_e, \quad \forall e \in E \quad (2) \\ \sum_{p: k(p)=k} \phi_p z_p^i \leq d_k, \quad \forall k \in \mathcal{K} \quad (3) \\ \phi_p \leq \phi_p z_p^i, \quad \forall p \in \mathcal{P} \quad (4) \\ \phi_p \geq 0, \quad \forall p \in \mathcal{P} \end{array} \right) \end{array} \right.$$

La fonction objectif consiste à maximiser la somme totale du flot porté par les chemins. La quantité $\alpha_e \sum_{p:e \in q} \phi_p$ représente le délai sur l'arc e qui est proportionnel à la quantité de flot transitant sur les chemins traversant cet arc. Pour chaque combinaison de chemins "actifs" nous exprimons les contraintes suivantes. Les contraintes (1) assurent que les chemins "actifs" qui supportent un flot non nul ne dépassent pas le délai maximum autorisé. Les contraintes (2) obligent le flot des chemins "actif" sur chaque liaison e à respecter la capacité de cette liaison. Les contraintes (3) imposent que la quantité maximum de flot pour la demande k sur les chemins "actifs" soit au plus la valeur d_k de la demande $k \in \mathcal{K}$. Nous constatons que si le chemin n'est pas actif ($z_p^i = 0$) alors les contraintes (1), (2) et (3) sont vérifiées. Donc la solution optimale est la combinaison de I permettant de transporter le plus de flot possible tout en respectant les contraintes (1), (2) et (3).

Nous avons donc pour chaque élément $i \in I$ une disjonction à laquelle nous associons un polyèdre $Poly(F_i)$ défini de la manière suivante.

$$Poly(F_i) = \left\{ \phi \in \mathbb{R}^{\mathcal{P}} \left| \begin{array}{l} z_q^i \left(\sum_{e \in q} \alpha_e \sum_{p:e \in p} \phi_p z_p^i \right) \leq 1, \quad \forall q \in \mathcal{P} \\ \sum_{p:e \in p} \phi_p z_p^i \leq C_e, \quad \forall e \in E \\ \sum_{p:k(p)=k} \phi_p z_p^i \leq d_k, \quad \forall k \in \mathcal{K} \\ \phi_p \leq \phi_p z_p^i, \quad \forall p \in \mathcal{P} \\ \phi_p \geq 0, \quad \forall p \in \mathcal{P}_i \end{array} \right. \right\}$$

L'ensemble I est fini (les $2^{|\mathcal{P}|}$ combinaisons possibles de l'ensemble des chemins) et chacun des polyèdres $Poly(F_i)$ pour $i \in I$ est borné. Cela nous permet d'affirmer que l'enveloppe convexe de $F = \bigcup_{i \in I} Poly(F_i)$ est le polyèdre suivant :

$$clconv F = \left\{ \phi \in \mathbb{R}^{\mathcal{P}} \left| \begin{array}{l} \phi = \sum_{i \in I} \phi^i \\ \sum_{e \in q} \alpha_e \sum_{p: e \in p} \phi_p^i \leq \lambda^i, \quad \forall q \in \mathcal{P}_i \quad \forall i \in I \\ \sum_{p: e \in p} \phi_p^i \leq C_e \lambda^i, \quad \forall e \in E \quad \forall i \in I \\ \sum_{p: k(p)=k} \phi_p^i \leq d_k \lambda^i, \quad \forall k \in \mathcal{K} \\ \sum_{i \in I} \lambda^i = 1, \\ \phi_p^i \geq 0, \quad \forall p \in \mathcal{P}_i \quad \forall i \in I \\ \lambda^i \geq 0, \quad \forall i \in I \end{array} \right. \right\}$$

Le Corollaire 2.3.1 nous permet d'affirmer qu'une solution optimale du programme disjonctif DP est la solution optimale du programme linéaire suivant.

$$F = \left\{ \begin{array}{l} Max \quad \sum_{i \in I} \sum_{p \in \mathcal{P}_i} \phi_p^i \\ s.c \\ \sum_{e \in q} \alpha_e \sum_{p: e \in p} \phi_p^i \leq \lambda^i, \quad \forall q \in \mathcal{P}_i \quad \forall i \in I \quad : y_q^i \quad (1) \\ \sum_{p: e \in p} \phi_p^i \leq C_e \lambda^i, \quad \forall e \in E \quad \forall i \in I \quad : \beta_e^i \quad (2) \\ \sum_{p: k(p)=k} \phi_p^i \leq d_k \lambda^i, \quad \forall k \in \mathcal{K} \quad \forall i \in I \quad : \delta_k \quad (3) \\ \sum_{i \in I} \lambda^i = 1, \quad : \gamma \quad (4) \\ \phi_p \geq 0, \quad \forall p \in \mathcal{P}_i \quad \forall i \in I \\ \lambda^i \geq 0, \quad \forall i \in I \end{array} \right.$$

La fonction objectif consiste à maximiser la somme totale du flot porté par les chemins. La contrainte (1) assure que les chemins supportant un flot respectent le

délai maximum. Les contraintes (2) sont les contraintes de capacité des arcs. Les contraintes (3) sont les contraintes de satisfactions des demandes. La contrainte (4) vient du fait que les λ_i décrivent une combinaison convexe.

Concrètement pour toute solution optimale du problème de flot à délai proportionnel il existe un indice $opt \in I$ tel que l'ensemble des chemins $\mathcal{P}_{opt}$ est le support de la solution optimal. Si nous fixons la variable

$$\lambda_i = \begin{cases} 1 & \text{si } i = opt \\ 0 & \text{sinon} \end{cases}$$

Ceci correspond à fixer l'ensemble des chemins utilisés. La solution optimale de F avec les λ_i fixés sera la solution optimale du problème de flot à délai proportionnel. Toute la difficulté de ce problème consiste donc à identifier un ensemble optimal $\mathcal{P}_{opt}$ parmi les $2^{|\mathcal{P}|}$ possibles. Le nombre de variables et de contraintes de ce programme linéaire est trop grand pour imaginer le construire explicitement. Nous désirons pouvoir construire $\mathcal{P}_{opt}$ de manière itérative en prenant en compte certaines informations. Afin d'obtenir ces informations nous décrivons le programme dual $\bar{F}$.

$$\bar{F} = \left\{ \begin{array}{l} Min \quad \gamma \\ s.c \\ \delta_{k(q)}^i + \sum_{e \in p} \left(\beta_e^i + \alpha_e \sum_{q: e \in q} y_q^i \right) \geq 1, \quad \forall p \in \mathcal{P}_i \quad \forall i \in I \quad : \phi_p^i \quad (1) \\ \gamma - \left(\sum_{k \in \mathcal{K}} d_k \delta_k^i + \sum_{q \in \mathcal{P}_i} y_q^i + \sum_{e \in E} C_e \beta_e^i \right) \geq 0, \quad \forall i \in I \quad : \lambda^i \quad (2) \\ y_q^i \geq 0, \quad \forall q \in \mathcal{P}_i \quad \forall i \in I \\ \beta_e^i \geq 0, \quad \forall e \in E \quad \forall i \in I \\ \delta_k^i \geq 0, \quad \forall k \in \mathcal{K} \quad \forall i \in I \\ \gamma \geq 0 \end{array} \right.$$

Ce programme linéaire nous permet d'obtenir des informations pour l'heuristique présentée dans la section suivante.

4.6 Heuristique par recherche locale

Le problème de flot à délai proportionnel étant NP-difficile nous proposons une heuristique de recherche locale permettant d'obtenir rapidement une solution réalisable. Cette méthode consiste à approximer le programme linéaire F défini par un nombre exponentielle de variables et de contraintes. La méthode de génération de colonnes 2.3 ne peut malheureusement pas s'appliquer à ce type de problème.

Pour le reste du chapitre nous définissons F_i comme le programme linéaire qui consiste à maximiser la somme totale du flot réalisable dans le polyèdre $Poly(F_i)$.

La méthode de résolution que nous employons consiste à explorer l'espace I des combinaisons de chemins actifs. Pour cela, nous choisissons un sous-ensemble initial de chemins et à chaque itération nous calculons la solution optimale ϕ pour le sous-ensemble de chemins courant. À partir des informations obtenues en résolvant le programme linéaire F_i courant, nous cherchons une modification élémentaire du sous-ensemble $\mathcal{P}_i$ qui conduit à une amélioration de la fonction objectif. Nous répétons de telles transformations tant que nous identifions un chemin dont l'ajout ou la suppression permet d'améliorer la solution courante. L'identification du chemin se base sur les observations suivantes :

Proposition 4.6.1. *Soit une combinaison de chemins $\mathcal{P}_i$ et F_i le programme linéaire associé. Si $\nexists q \in \mathcal{P} - \mathcal{P}_i$ tel que q a un coût réduit positif. Alors $\nexists j$ tel que $\mathcal{P}_j \supseteq \mathcal{P}_i$ et $opt(F_j) > opt(F_i)$.*

Démonstration 4.6.1.

Nous notons $F_i \oplus r$ le programme linéaire F_i avec l'ajout d'une variable r sans sa contrainte de chemin. Admettons qu'il existe $j \in I$, $\mathcal{P}_j = \mathcal{P}_i \cup q$ tel que $opt(F_j) > opt(F_i)$. $F_i \oplus r$ est au moins égale à $opt(F_k)$ avec $P_k = \mathcal{P}_i \cup r$ qui lui prend en compte la contrainte de chemin de r . Donc $opt(F_i \oplus r) \geq opt(F_i)$. Si q à un coût réduit négatif alors $opt(F_i) \geq opt(F_i \oplus q) \geq opt(F_j)$. Ce qui est contradictoire avec l'hypothèse initiale, donc si l'ajout de q à l'ensemble des chemins augmente la solution, cela implique que q a un coût réduit positif.

Nous pouvons appliquer ce même raisonnement pour un ensemble de chemin. S'il n'existe pas de chemin $q \in \mathcal{P} - \mathcal{P}_i$ ayant un coût réduit positif alors il n'existe pas d'ensemble de chemins $\mathcal{P}' \subseteq \mathcal{P} - \mathcal{P}_i$ tel $opt(F_i \oplus \mathcal{P}') \geq opt(F_i)$.

□

La première étape de notre algorithme consiste à créer un ensemble de chemins de manière itérative en identifiant à un instant t le chemin dont la variable à un coût réduit positif, donc étant susceptible d'augmenter la valeur de la fonction objectif. Identifier une telle variable revient à trouver une contrainte du dual violée (condition suffisante). Nous considérons $(\beta_e : e \in E, y_q : q \in P, \delta_k : k \in \mathcal{K})$ la solution optimale du programme linéaire dual courant $\bar{F}$. Pour chaque demande $k \in \mathcal{K}$ nous recherchons un plus court chemin entre s_k et t_k avec une pondération $\left(\beta_e + \alpha_e \sum_{q: e \in p} y_q \right)$ pour tout arc $e \in E$, si la longueur du plus court chemin est strictement inférieure à $1 - \delta_k$ alors la variable associée à ce plus court chemin a un coût réduit positif.

Proposition 4.6.2. *Si dans une solution optimale de F_i une variable ϕ_q a une valeur nulle alors $\text{opt}(F_j) \geq \text{opt}(F_i)$ avec $\mathcal{P}_j = \mathcal{P}_i - q$.*

Cette proposition nous permet d'identifier des chemins dont la suppression ne dégrade pas la valeur de la fonction objectif.

Nous proposons donc l'algorithme 7 pour résoudre de manière heuristique le problème de flot à délai proportionnel.

Notre heuristique est un algorithme de recherche locale dans l'espace des combinaisons de chemins. L'évaluation de la fonction objectif pour la combinaison de chemin courante se fait en résolvant un programme linéaire. Le choix du déplacement est basée sur les informations associées à la base optimale de ce programme linéaire. Cette procédure ne garantit pas d'obtenir une amélioration de la fonction objectif car l'ajout de la contrainte associé au chemin ajouté peut avoir un effet pénalisant. Cette variable correspond à l'ajout d'un chemin de coût réduit positif. Grâce à notre modélisation en programme linéaire nous pouvons certifier que lorsque l'on ne trouve plus de chemin à coût réduit positif, la combinaison de chemins courante est au moins aussi bonne que toute combinaison incluant celle-ci. Dans ce cas notre heuristique supprime les variables de valeurs nulles. Ce sont des chemins inutiles car ils ne portent pas de flot mais la contrainte sur ces chemins est active (donc surcontraignant le programme linéaire courant). Afin d'éviter de rester bloqué dans un "minimum" local nous interdisons d'ajouter un chemin qui vient juste d'être supprimé. Pour cela nous utilisons T une liste de chemins tabous.

Algorithme 7: Heuristique pour le problème de flot avec délais proportionnel

Données : $G = (V, E)$; $C \subset \mathbb{R}_+^{E(G)}$; $\alpha \subset \mathbb{N}_+^{E(G)}$; $\mathcal{K} \subset \mathbb{N}_+^{E^2(G)}$; $itMax \in \mathbb{N}_+$

Résultat : $(\phi_p : p \in P)$ est une solution pour le problème de flot à délai proportionnel

```

1   $T = NULL$  // Chemin Tabou;
2   $i = \emptyset$  // L'ensemble de chemins associé est  $\mathcal{P}_i$ ;
3  tant que le nombre d'itération est inférieure à  $itMax$  faire
4      si  $(\exists q$  chemin dont le coût réduit est positif) et  $(q \neq T)$  alors
5          // Nous ajoutons un chemin ;
6           $j : \mathcal{P}_j = \mathcal{P}_i \cup q$ ;
7           $i = j$  ;
8          Résoudre le programme linéaire  $F_i$  ;
9      fin
10     sinon
11         // Nous supprimons les chemins ne supportant pas de flot ;
12         tant que  $\exists q$  chemin ne supportant pas de flot faire
13              $T = q$  ;
14              $j : \mathcal{P}_j = \mathcal{P}_i - q$ ;
15              $i = j$  ;
16             Résoudre le programme linéaire  $F_i$  ;
17         fin
18         si Aucun chemin a été supprimé alors
19             On sort de la boucle ;
20         fin
21 fin

```

21 Retourner $(\phi_p : p \in \mathcal{P}_i)$ la valeur de la solution optimale de F_i .

Simulations

Dans cette section nous présentons succinctement les tests expérimentaux effectués. Nous avons généré des instances de problème de flot avec délai borné issues du problème Partition pour lesquelles nous sommes en mesure de calculer une solution optimale. Par la suite nous appelons les instances partition cette famille d'instances. La deuxième famille d'instances étudiées sont les instances pour lesquelles le graphe est un chemin que nous appellerons par la suite instance chemin.

L'algorithme est implémenté en JAVA et nous utilisons le solveur libre GLPK⁵ pour la résolution des programmes linéaires.

Instances partition : La génération des instances du problème Partition prend deux paramètres : le nombre total d'éléments et la valeur d'une des parties. Pour déterminer le nombre d'éléments de la première partie nous tirons aléatoirement un entier compris entre un quart et un demi du nombre total d'éléments afin de ne pas avoir des partitions trop déséquilibrées en terme de nombre d'éléments. Le tirage aléatoire des poids des éléments d'une partie se fait de la façon suivante : nous calculons le poids moyen de cette partie, ensuite pour chaque élément nous tirons aléatoirement un poids que nous additionnons ou soustrayons si la moyenne courante est inférieure ou supérieure à la moyenne espérée, pour finir le dernier élément de la partie est dépendant des autres éléments car il équilibre la valeur de la partie pour que la somme totale des éléments soit celle désirée. Nous proposons les résultats expérimentaux obtenues pour les instances du problème de partition sous la forme du tableau 4.1 où chaque ligne nommée *NbElement_ValPartie* correspond à la moyenne de 100 instances caractérisées par les paramètres de nombre d'éléments et de la valeur des parties. Le gap d'optimalité représente la différence entre la valeur optimale et la valeur renvoyée par l'algorithme par rapport à la solution optimale.

Nous remarquons que la valeur de la partition n'influence pas les résultats. Le nombre de solutions optimales n'est pas dépendant des paramètres de l'instance. De très bonnes solutions sont obtenues et le nombre d'itérations croît de façon sous linéaire par rapport à la taille de l'instance. Les tests expérimentaux montrent une convergence rapide vers une très bonne solution jusqu'à atteindre un minimum local.

5. version 4.45 : <http://www.gnu.org/software/glpk/>

4.6. HEURISTIQUE PAR RECHERCHE LOCALE

instance	$\frac{iteration}{NbElement}$	NbOpti	Gap opti
10_80	5.2	75	1.15 %
50_80	4.2	7	8.06 %
100_800	4.93	78	0.46 %
100_8000	4.9	69	0.69 %
100_80000	4.88	72	0.71 %
150_8000	4.22	6	7.69 %
200_8000	4.18	9	8.12 %
250_8000	4.24	13	7.45 %
300_8000	4.92	75	8.89 %
350_8000	4.28	11	7.45 %

TABLE 4.1 – Résultats des instances du problème partition

Instances du graphe chemin : La génération d’une instance chemin prend en entrée deux paramètres : le nombre de sommets du chemin et le nombre de demandes. Les instances chemins présentent un avantage. Nous appliquons une étape de pré-processing qui consiste à supprimer les demandes qui sont complètement “dominées” par une plus petite, c’est à dire que si pour deux demandes $k_1, k_2 \in \mathcal{K}$ le sous-chemin reliant la source et la destination de k_1 est inclus dans le sous-chemin source-destination de k_2 alors nous pouvons supprimer k_2 sans détériorer la valeur de l’optimum. Ce prè-processing permet de réduire considérablement le nombre de demandes finales car le nombre de demandes “non dominées” au plus le nombre de sommets.

Le tableau 4.2 décrit les résultats obtenues, chaque ligne nommée *NbElement_NbDemande* correspond à la moyenne de 100 instances caractérisées par les paramètres du nombre d’éléments et du nombre de demandes. Nous comparons deux stratégies d’ensemble de chemins initial nommées *Algo1* et *Algo2*, la première étant comme pour les instances précédentes de partir d’un ensemble vide et la seconde étant de partir avec tous les chemins dont la taille est égale au nombre de demandes. Le pourcentage calculé pour nos deux algorithmes se fait par rapport à une valeur de la solution de notre algorithme de 2-approximation. Nous considérons la valeur de l’approximation comme minimale.

instance	$\frac{iteration1}{NbDemande}$	% ₁	$\frac{iteration2}{NbDemande}$	% ₂
50_50	1.07	103.46	0.24	103.14
100_100	1.05	103.25	0.19	102.5
200_200	1.02	102.28	0.14	102.0
300_300	1.02	102.29	0.12	102.03
400_400	1.01	101.89	0.12	101.79
500_500	1.0	101.94	0.11	101.8
5000_50	1.05	100.34	0.12	100.33
5000_500	1.01	100.81	0.07	100.8
5000_2500	0.99	100.91	0.05	100.89
5000_5000	0.98	101.12	0.05	101.08

TABLE 4.2 – Résultats sur les instances chemin

Nous mettons en évidence dans ce tableau que l'ensemble initial de chemins influence peu le résultat final mais permet de converger plus vite pour le cas de l'*Algo2*. Nous observons que l'heuristique améliore peu la solution par rapport à l'algorithme d'approximation. Nous constatons que notre heuristique atteint rapidement des minimums locaux. À ce jour nous ne savons pas si le problème est NP-difficile sur le graphe chemin.

4.7 Conclusions

La recherche de réseaux de communications garantissant un niveau de qualité de service motive l'étude de la question du délai de transmission de l'information dans ces réseaux. La prise en compte du délai pour les problèmes de dimensionnement de réseaux a conduit à généraliser le problème de flot en lui ajoutant une contrainte de délai. Les mesures de délai effectuées sur les réseaux mères des opérateurs mettent en avant une relation entre le taux d'utilisation d'une liaison de transmission et le délai. Celle-ci se présente sous la forme d'une fonction convexe croissante. Des travaux ont déjà été réalisés où l'on autorise qu'un sous-ensemble fixe de chemins [85], mais aucune solution n'a été proposée pour le cas général du problème de flot avec contrainte de délai.

Dans un premier temps, nous montrons que la difficulté du problème n'est pas liée à la convexité de la fonction de délai. Nous apportons la preuve que ce problème reste NP-difficile dans le cas où le délai est proportionnel. Nous proposons deux preuves de NP-complétude pour des spécifications du problème de flot avec

contrainte de délai proportionnel respectivement pour le problème sans capacité et pour le problème sans demande. Nous proposons pour le cas simple du graphe chemin sans capacité avec des demandes un algorithme d'approximation facteur 2. Finalement nous proposons une heuristique de recherche locale pour le cas général avec délai linéaire en utilisant une modélisation basée sur la programmation disjonctive. Ceci nous permet d'explorer l'espace des combinaisons de chemins en construisant un ensemble de chemins supportant le flot, que nous améliorons en ajoutant ou supprimant des chemins. Nous présentons des résultats expérimentaux sur des instances du problème de partition et des instances du problème sur le graphe chemin.

Parmi les perspectives envisagées à la suite de ce travail nous pouvons mentionner la recherche d'un algorithme qui affine l'approximation des fonctions de délai convexes de manière itérative. Cet algorithme consisterait à toujours garder une solution réalisable et à affiner dynamiquement l'approximation de la fonction convexe de délai en fonction de la solution courante. Différentes stratégies pourraient être étudiées. Premièrement, si nous supposons que la famille de chemins supportant le flot est identique entre deux itérations d'approximation. Dans ce cas, considérons une solution optimale dont nous souhaitons affiner l'approximation de la fonction de délai sur l'arc e . Soit ϕ_e le flot supporté par un arc e , D_e son délai et F_e la fonction convexe de délai de l'arc e . Nous pouvons supposer que le flot optimal après l'approximation ϕ_e^* est supérieur ou égal à ϕ_e , car une meilleure approximation du délai relâche les contraintes de délai sur les chemins traversant e et par conséquent autorise plus de flot sur cet arc. Deuxièmement, nous pouvons supposer que ϕ_e^* est inférieur ou égal à $F_e^{-1}(D_e)$ le flot maximum sur l'arc e tel que le délai D_e soit atteint par la fonction de délai convexe. En prenant en compte ces deux supposition, nous pouvons espérer améliorer l'approximation en créant un ensemble de fonctions linéaires approximant la fonction de délai convexe, tel que les points $(\phi_e, F_e(\phi_e))$ et $(F_e^{-1}(D_e), D_e)$ satisfassent à l'égalité deux fonctions linéaires consécutives. Ces points correspondent aux points de cassure de la fonction linéaire par morceaux (définie par l'ensemble des fonctions linéaires) approximant la fonction convexe.

Protocoles pour les réseaux de capteurs

5

Résumé

Les réseaux de capteurs et leurs applications soulèvent de nombreuses questions algorithmiques. Dans ce chapitre, nous étudions deux problèmes particuliers qui concernent le choix des chemins de routage des communications à travers un réseau de capteurs. Ces problèmes nous ont été soumis par une équipe de Orange R&D qui travaille sur le déploiement de capteurs dans les villes.

Le premier problème consiste à rechercher un routage des communications qui optimise la résistance aux pannes du réseau. La solution que nous proposons consiste à faire en sorte que deux communications différentes n'empruntent jamais la même liaison. En respectant cette contrainte, on assure qu'une panne sur une liaison affecte au plus une communication. La recherche des routages se fait au moyen d'un algorithme distribué qui calcule un ensemble maximum de chemins arêtes-disjoints entre un ensemble de terminaux du réseau. Le procédé développé a fait l'objet d'un dépôt de brevet international [91].

Le second problème traite de la question de la durée de vie d'un réseau de capteurs. Les capteurs fonctionnant pour la plupart sur batterie, la gestion de leur durée de vie est importante. Nous proposons un protocole de routage permettant de maximiser la durée de vie d'un réseau de capteurs [90].

5.1 Contexte

Dans cette section, nous présentons le contexte de notre étude. Dans un premier temps nous décrivons la composition d'un capteur et son champ d'application. Ensuite nous aborderons les différents types de réseaux de capteurs et les problématiques associées.

Les capteurs

Un capteur est un système embarqué composé d'un micro-contrôleur, d'un émetteur-récepteur, d'une mémoire externe, d'une source d'alimentation et d'un ou plusieurs transducteurs. Le micro-contrôleur est l'élément de calcul qui commande le fonctionnement des autres composants du capteur. Le transducteur convertit une mesure physique comme la température, la vitesse, la photométrie, etc . . . en un signal électrique. Ce signal est transmis et interprété par le micro-contrôleur. La plupart du temps la source d'alimentation est une batterie. La multiplicité de type de capteurs nous conduit à considérer simplement un capteur comme un dispositif capable de fournir une information relative à son environnement.

Les progrès réalisés ces vingt dernières années dans le domaine de la micro-électronique ont permis la réalisation de capteurs suffisamment petits et surtout suffisamment peu onéreux pour que leur utilisation en grand nombre soit économiquement possible. Ceci a conduit à la création de grands parcs de capteurs. Il a donc été nécessaire de bâtir des réseaux susceptibles de récolter les informations fournies par les capteurs. La construction d'un réseau de capteurs vient du fait que ces éléments peuvent être considérés à la fois comme émetteur et récepteur permettant ainsi de router l'information. Les capteurs peuvent donc être considérés comme des éléments constitutifs du réseau.

Les capteurs actuels sont capables de mesurer différentes grandeurs physiques ou chimiques. Ceci permet de donner, par exemple, de l'information sur la position d'un objet à la surface de la terre, la température ou le niveau d'eau, niveau de lumière, présence de certains corps chimiques dans l'air, etc . . . Les plus sophistiqués sont même capables de fournir des images. Ces diverses possibilités leur assurent de nombreux domaines d'applications.

Les Applications

Les domaines actuels dans lesquels interviennent les réseaux de capteurs peuvent être regroupés de la manière suivante.

La sécurité : Il s'agit de capteurs qui ont pour but de détecter les paramètres relatifs à une situation anormale. Dans les bâtiments ou les infrastructures, ils servent à détecter l'intrusion d'une personne ou la dégradation d'une installation. Dans un environnement plus vaste, ils peuvent remonter une alerte météorologique, détecter de façon précoce des incendies de forêt ou des séismes. Les réseaux de sismographes sont parmi les plus anciens réseaux de capteurs.

La surveillance de l'environnement : Dans ce domaine les capteurs ont pour but de fournir des informations sur les paramètres écologiques de l'environnement. Que ce soit l'analyse de l'air ou de l'eau, la demande des collectivités locales pour ce type de réseau est forte. Cela leur permet par exemple de gérer l'allumage des lampadaires, l'arrosage automatique des espaces verts publics, la prise en compte de l'impacte de la pollution atmosphérique des axes routiers ou simplement informer les touristes de façon précise de la température de l'eau en période estivale. Les applications agricoles peuvent être rangées dans ce domaine, elles permettent de déterminer les assolements optimaux pour les différentes cultures.

Les applications relatives à la chaîne support logistique : Ici les capteurs doivent délivrer des informations sur un véhicule, nous parlons alors de gestion de flotte, ou une marchandise. Ceci permet d'améliorer le processus de stockage, par exemple, pour garantir la chaîne du froid de produits surgelés, mais il est aussi possible de les utiliser pour contrôler un processus de livraison, connaître la position, l'état et la direction d'un paquet ou d'une cargaison. Par exemple, un client attendant un paquet peut alors savoir, en temps réel, l'état et la position du colis.

Le domaine militaire : Comme pour ce qui est de la sécurité le but est de surveiller un champ d'opérations ainsi que les actions logistiques qui doivent subvenir aux besoins des opérations. Parmi leurs différents types d'utilisation de ces réseaux militaires nous trouvons, par exemple, la surveillance de mouvements, l'analyse du terrain avant d'y envoyer des troupes avec la détection d'agents chimiques, biologiques ou de radiations. Le domaine militaire est le principal moteur de l'évolution et du développement des réseaux de capteurs. Les principaux axes de recherche s'attachent à obtenir un déploiement rapide, un coût réduit, une capacité d'auto-organisation et de tolérance aux pannes.

La domotique et le domaine médical : Les applications au niveau de l'habitat passent par la simple mesure de la luminosité pour commander les stores jusqu'à

l'automatisation de tâches ménagères, mais aussi la détection de gaz dangereux comme le monoxyde de carbone. Dans le domaine médical, la surveillance des fonctions vitales d'un organisme vivant est maintenant améliorée par des micro-capteurs ingérés ou implantés sous la peau.

Les réseaux de capteurs

Les réseaux de capteurs sans fil sont basés sur une technologie émergente [4]. Un réseau de capteurs se compose d'éléments distribués qui peuvent, dans certains cas, s'organiser eux-mêmes dans le réseau. Lorsque les éléments sont mobiles et peuvent apparaître et disparaître, on parle alors de réseaux *Ad-hoc*. Les communications dans les réseaux de capteurs sans fils s'effectuent, comme pour les réseaux *Ad-hoc*, par bond. Une communication *multi-bond* (multi-hop) consiste à faire transiter une donnée par un ou plusieurs éléments intermédiaires lorsque la portée de télécommunication ne permet pas un échange direct de l'information.

Pour obtenir un réseau de capteurs à la fois durable et évolutif, les aspects suivants doivent être considérés lors de sa phase de conception. Les communications sont de faibles qualités et la bande passante est limitée. Le traitement des données est restreint par la puissance de calcul des capteurs. La taille des données peut aussi être un facteur limitant. L'environnement extérieur est considéré comme hostile et peut imposer une topologie de réseau variable dans le temps.

Plusieurs solutions ont été proposées dans la littérature pour prendre en compte ces critères. En particulier, des efforts considérables ont été consacrés à la gestion de l'énergie et à la transmission efficace des données.

Nous modélisons les réseaux de capteurs par un graphe ou un digraphe. Chaque sommet du graphe représente un capteur. Une arête du graphe symbolise une communication possible entre les sommets. Le réseau est souvent supposé *homogène*, c'est-à-dire que tous les sommets possèdent les mêmes attributs de réseau tels que la capacité de calcul, la puissance de la batterie.

Les protocoles de communications dans ce type de réseau comportent souvent deux étapes. La première consiste à construire la topologie du réseau. La seconde calcule un routage dans cette topologie.

L'arbre couvrant de poids minimum et les ensembles de chemins disjoints sont des notions souvent utilisées pour la construction de la topologie. La première et ses variantes favorise le critère d'économie d'énergie. La seconde intervient fréquemment lorsque l'objectif consiste à obtenir un réseau tolérant les pannes.

5.2 Routage sûr de l'information dans les réseaux de capteurs

Dans cette section, nous considérons des réseaux de capteurs ayant un système critique de transmissions de l'information. Dans ce type de système, une panne peut avoir des conséquences très négatives.

Dans notre cas, nous supposons que la communication entre les capteurs est soumise aux pannes. Elle peut être altérée par un événement extérieur qui bloque la transmission se traduisant par une panne de lien dans le réseau.

Actuellement les capteurs mesurent un paramètre du milieu dans lequel ils sont implantés et transmettent cette donnée vers un point où elle sera traitée. À terme, les réseaux de capteurs seront capables d'agir, soit seul, soit en association avec d'autres dispositifs semblables. Les applications de ces types de réseaux critiques sont par exemple : des mécanismes QoS en temps réel, la gestion avancée du trafic, la mise en place de techniques d'analyse d'image pour les appareils limités en énergie, l'analyse de scène 3D à partir de capteurs d'images distribués ou bien la mise en place d'algorithmes optimisés pour les systèmes de détection d'intrusion.

Sécurité des communications

Nous proposons un protocole de routage permettant à un ensemble de capteurs de communiquer entre eux. Le graphe est dynamique et évolue dans le temps, mais il est supposé fixe durant la recherche du routage. Les communications asynchrones entre les capteurs ne sont pas soumises aux interférences dû à la concurrence des communications (elles utilisent un protocole de transport de type WiFi pour les couches liaison et physique du modèle OSI). Nous supposons que les capteurs disposent d'une mémoire tampon leur permettant de traiter les messages par ordre d'arrivée.

Pour assurer la sécurité de la transmission nous souhaitons pouvoir faire transiter les données entre deux capteurs par des chemins disjoints, au sens des arêtes, c'est-à-dire que les ensembles de liaisons qu'ils utilisent sont disjoints. Notre modèle ne prend donc en compte que les pannes de liaisons. Notre problème consiste donc à trouver un ensemble de routes disjointes pour transmettre l'information. Nous considérons la taille de cet ensemble comme critère d'évaluation de la solution calculée. Le problème que nous considérons peut être formulé de la façon suivante :

Problème: Routage chemins-disjoints

Entrée: Un graphe $G = (V, E)$, un sous ensemble T de sommets désirant communiquer

Question: Trouver l'ensemble maximal de chemins arêtes disjoints reliant les sommets de T .

La recherche d'une topologie virtuelle visant à maximiser le nombre total de chemins arêtes-disjoints est un problème qui a déjà été étudié [15, 44, 143]. Mais, à notre connaissance, les travaux existant traitent seulement le cas où deux capteurs souhaitent communiquer entre eux. Étant donné un sous-ensemble de sommets d'un graphe, le problème de trouver le nombre maximum de chemins disjoints par les arêtes est un problème classique en optimisation combinatoire [137, 138]. Le théorème de Mader définit une relation entre le nombre maximum de chemins disjoints par les arêtes dans un graphe et une valeur associée à une partition des sommets. Il existe un algorithme polynomial basé sur la résolution d'un programme linéaire pour résoudre ce problème [102], mais aucun algorithme polynomial combinatoire n'est connu pour ce problème. Nous proposons un algorithme distribué qui permet d'obtenir une solution sans garantie théorique de performance. Les simulations que nous avons menées montrent que cet algorithme produit de bonnes solutions pour les instances testées.

Remarques : Une autre problématique consisterait à spécifier le nombre de chemins de routage que l'on souhaite mettre en place entre chaque paire de terminaux. Ce problème correspond à un problème de multiflot entier, il devient alors très difficile à résoudre [59] et à approximer [81] même dans un contexte non-distribué.

Protocole de routage

Le réseau est représenté par un graphe $G = (V, E)$ dont les sommets représentent les capteurs et les arêtes indiquent une communication possible entre deux capteurs. Nous nommons *terminaux* le sous-ensemble $T \subseteq V$ des sommets désirant communiquer entre eux. La communication se faisant par bonds, nous interdisons à un chemin de routage joignant deux terminaux t et t' de passer par un autre terminal de T . Par ailleurs nous supposons que les sommets du graphe n'ont pas une connaissance générale du graphe mais connaissent leurs voisins. Les chemins étant disjoints, un chemin ne peut emprunter une arête déjà utilisée pour router une communication. On note $\mathcal{P}$ l'ensemble des chemins (routes) entre les terminaux.

Comme mentionné précédemment nous recherchons des chemins disjoints par les

arêtes pour des raisons de sûreté de fonctionnement du réseau. Pour cela chaque terminal t va étendre sa zone d'influence qui est une composante connexe du graphe contenant t . Notre protocole assure que les zones d'influence (ou *clusters*) des différents terminaux sont des composantes disjointes, c'est à dire qu'un sommet ne peut pas appartenir à plusieurs clusters.

Le protocole que nous proposons est basé sur l'algorithme 8 qui représente de manière centralisé la recherche d'un routage disjoint des communications entre un ensemble de composant du réseau. Nous notons X_t les sommets du cluster du terminal t construit par l'algorithme. Nous notons B_t l'ensemble des voisins du domaine X_t qui ne sont dans aucun domaine. Quand deux sommets voisins u, v appartiennent à deux clusters différents $u \in X_t$ et $v \in X_{t'}$, nous essayons d'établir un routage entre t et t' . Nous définissons $\mathcal{P}_G(t, u)$ l'ensemble des chemins reliant le sommet t au sommet u dans le graphe $G[X_t]$. Le routage est obtenu par concaténation d'un chemin $q \in \mathcal{P}_G(t, u)$, de l'arête $\{u, v\}$ et d'un chemin $q' \in \mathcal{P}_G(v, t')$. La route $p = q \cdot uv \cdot q'$ ainsi créée est ajoutée à $\mathcal{P}$ peut échouer. La recherche d'un chemin entre un sommet $v \in X_t$ et son terminal t . Lorsqu'une route est choisie les arêtes qui appartiennent à celles-ci ne sont plus disponibles pour d'autres routes et sont donc supprimées. Cette suppression peut avoir pour conséquence de rendre un sommet $v \in X_t$ non accessible à partir de t . Dans ce cas, nous en concluons que le sommet doit être enlever de l'ensemble X_t . La recherche des chemins $\mathcal{P}_G(t, u)$ s'effectue seulement dans le cluster X_t . Nous notons $E(p)$ l'ensemble des arêtes du chemin p .

Discussion sur l'heuristique. Une heuristique semblable à celle que nous décrivons pour résoudre ce problème de routage consisterait à calculer à chaque itération la distance entre toutes les paires de terminaux, à ajouter à $\mathcal{P}$ le plus court chemin entre les deux terminaux les plus proches et à supprimer les arêtes de ce chemin pour les itérations suivantes. Les simulations effectuées montrent que les solutions renvoyées par cette heuristique sont très proches des solutions renvoyées par celle que nous décrivons ci-dessous. L'avantage de l'algorithme que nous proposons est qu'il est plus simple à implémenter de manière distribuée dans un réseau ayant des communications asynchrones avec des capteurs ayant peu de ressources mémoires et une connaissance limitée (locale) du réseau. Même s'il existe des algorithmes distribués pour le calcul des distances entre chaque paire de terminaux, avec un modèle asynchrone ayant des capteurs de faible ressources mémoires, la détection de la terminaison de tels algorithmes pose des problèmes et elle est nécessaire avant de passer à la phase suivante de sélection du plus court chemin.

Algorithme 8: Algorithme centralisé de routage sûr dans les réseaux critiques de capteurs

Données : $G = (V, E)$; $T \subseteq V$
Résultat : Ensemble de routes disjointes dans le réseau

```

1   $X_t = \{t\}, \forall t \in T$  ;
2   $\mathcal{P} = \emptyset$  ;
3   $G' = G$  ;
4  tant que  $\exists x \in V - T$  tel que  $x \notin \bigcup_{t \in T} X_t$  faire
5       $B_t = \emptyset, \forall t \in T$  ;
6      pour tous les  $u \in \bigcup_{t \in T} X_t$  faire
7          si  $\exists v$  voisin de  $u$  dans  $G'$  tel que  $v \in X_t$  et  $u \notin \bigcup_{t' \in T} B_{t'}$  alors
8               $B_t = B_t \cup u$  ;
9          fin
10     fin
11      $X_t = X_t \cup B_t, \forall t \in T$  ;
12     pour tous les  $uv \in E(G')$  tel que  $u \in X_t$  et  $v \in X_{t'}$  faire
13         si  $\mathcal{P}_{G'}(t, u) \neq \emptyset$  et  $\mathcal{P}_{G'}(v, t') \neq \emptyset$  alors
14             Soient  $q \in \mathcal{P}_{G'}(t, u)$  et  $q' \in \mathcal{P}_{G'}(v, t')$ ,  $p = q \cdot uv \cdot q'$  ;
15              $\mathcal{P} = \mathcal{P} \cup p$  ;
16              $E(G') = E(G') - E(p)$  ;
17         fin
18     sinon
19         si  $\mathcal{P}_{G'}(t, u) = \emptyset$  alors
20              $X_t = X_t \cap$  composante connexe de  $u$  dans  $G'$  ;
21         fin
22         si  $\mathcal{P}_{G'}(v, t') = \emptyset$  alors
23              $X_{t'} = X_{t'} \cap$  composante connexe de  $v$  dans  $G'$  ;
24         fin
25     fin
26 fin
27 fin
28 Retourner  $\mathcal{P}$ ;
```

5.2. ROUTAGE SÛR DE L'INFORMATION DANS LES RÉSEAUX DE CAPTEURS

Les figures suivantes illustrent le déroulement du processus. Sur la figure 5.1 nous pouvons visualiser le réseau sous-jacent et les quatre terminaux t_1 , t_2 , t_3 , t_4 qui souhaitent communiquer.

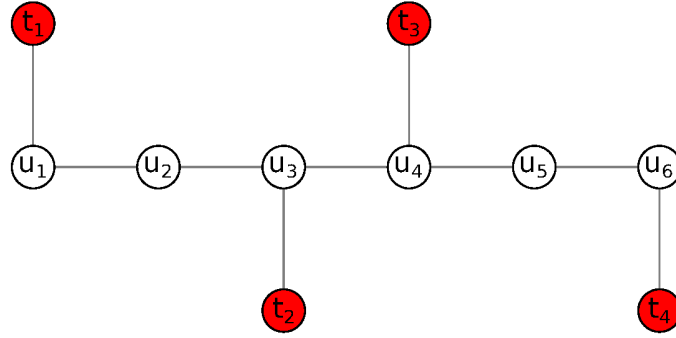


FIGURE 5.1 – Quatre terminaux désirant communiquer

Les clusters obtenues après la première itération de l'algorithme sont décrits sur la figure 5.2. Chacun est constitué de la boule de rayon 1 centré dans le terminal correspondant.

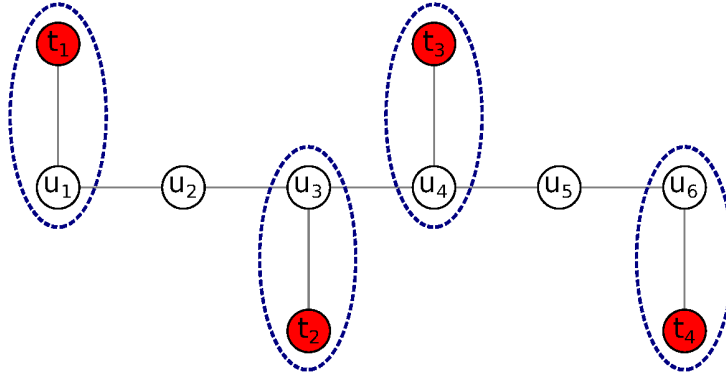


FIGURE 5.2 – Accroissement des clusters

Lors de la première itération, l'arête u_3u_4 appartenant à deux clusters différents apparaît en gras sur la figure 5.3. Cette arête est sélectionnée lors de la première itération. Ainsi un chemin est trouvé, il est constitué du chemin entre u_3 et t_2 de l'arête u_3u_4 et du chemin entre u_4 et t_3 . Les chemins de u_3 vers t_2 et de u_4 vers t_3 en pointillés sur la figure 5.3.

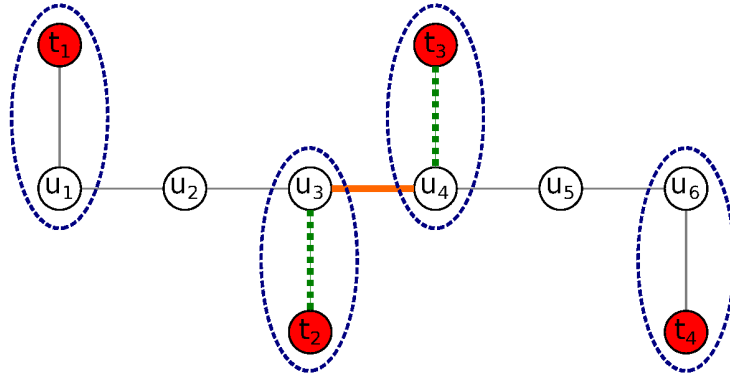


FIGURE 5.3 – Chemin trouvé

Les arêtes du chemin sont retirées du graphe. Nous obtenons alors le graphe résiduel G' sur lequel l'algorithme continue à itérer. Le graphe obtenu après cette suppression est dessiné sur la figure 5.4.

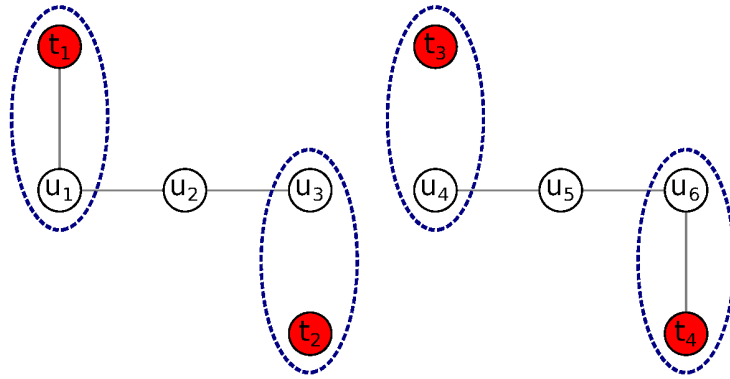


FIGURE 5.4 – Graphe résiduel G' après suppression des arêtes des chemins sélectionnés

La figure 5.5, représente l'évolution des clusters pour la seconde itération, ainsi que les arêtes sélectionnées u_2u_3 et u_5u_6 .

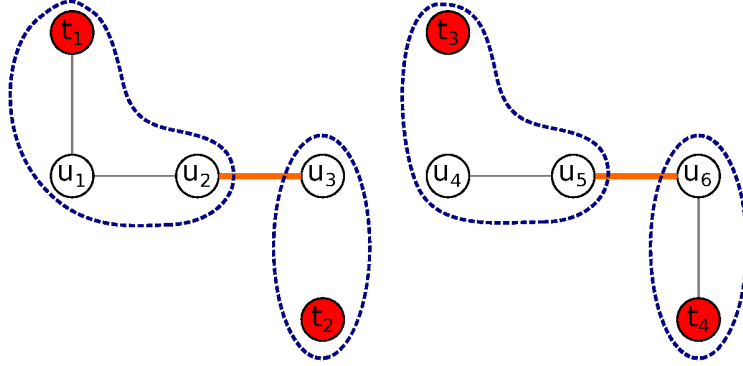


FIGURE 5.5 – Évolution de l'algorithme, seconde itération

Trouver un chemins reliant u_3 à t_2 et un autre reliant u_5 à t_3 est impossible. Donc la composante connexe de u_3 est retirée de X_{t_2} et la composante connexe de u_5 est retirée de X_{t_3} . On obtient donc le graphe résiduel G' représenté par la figure 5.6.

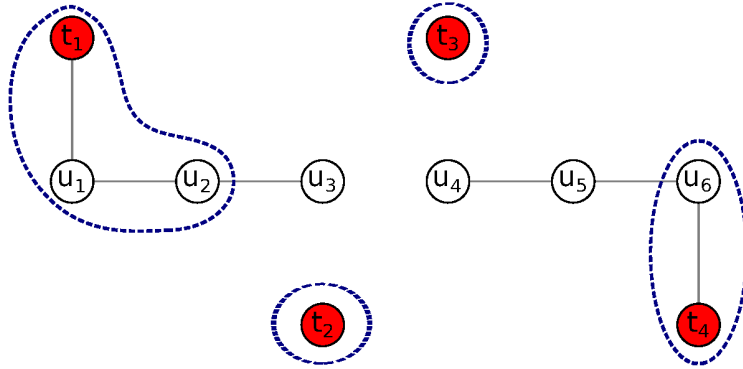


FIGURE 5.6 – Évolution de l'algorithme, troisième itération

À la fin de l'algorithme, X_{t_1} contient les sommets $\{t_1, u_1, u_2, u_3\}$, X_{t_4} contient les sommets $\{t_4, u_4, u_5, u_6\}$ et les clusters X_{t_2} et X_{t_3} contiennent uniquement leur terminal.

L'algorithme retourne un chemin comme solution. Pour cet exemple notre algorithme ne renvoie pas une solution optimale. La solution optimale est constituée de deux chemins, celui reliant t_1 à t_2 et celui reliant t_3 à t_4 .

Analyse de l'algorithme 8

Il est clair que l'algorithme renvoie une solution. L'ensemble $\mathcal{P}$ définit un ensemble de chemins. Ces chemins sont disjoints car dès l'ajout d'un chemin p dans $\mathcal{P}$, les arêtes de p sont immédiatement retirées du graphe, donc aucun autre chemin ne peut les utiliser.

Une itération de l'algorithme comporte trois phases. La première phase consiste à faire grossir les clusters en leur ajoutant leur frontière après l'avoir calculée. Il suffit pour cela de parcourir la liste des voisins de chaque sommet appartenant à un cluster. Cette phase nécessite donc $O(|E|)$ opérations. La deuxième phase consiste à identifier les paires de sommets voisins appartenant à deux clusters différents. De nouveau, il suffit pour cela de parcourir la liste des voisins de chaque sommet appartenant à un cluster ce qui nécessite aussi $O(|E|)$ opérations. Pour finir, la troisième phase de l'itération consiste, pour chaque paire de sommets voisins uv appartenant à deux clusters X_t et $X_{t'}$ distincts (identifiés lors de la phase précédente), à essayer de construire un chemin entre t et t' passant par l'arête uv . Cette phase nécessite un parcours de graphe en $O(|E|)$ pour déterminer si u et v sont atteignables respectivement à partir de t et de t' dans le graphe G' . Si ce n'est pas le cas, par exemple, u n'est pas atteignable à partir de t alors une mise à jour de X_t est effectuée, celle-ci nécessitant de nouveau $O(|E|)$ opérations. Si au contraire un chemin est identifié entre t et t' les arêtes de ce chemin sont retirées du graphe G' avant de commencer une nouvelle itération. Au total une itération de l'algorithme coûte $O(|E|)$ opérations.

A chaque itération, si aucun chemin n'est ajoutée au moins un cluster grossit. Par conséquent entre deux itérations où un chemin est ajouté, nous avons au maximum $O(|V|)$ itérations où aucun chemin n'est ajouté. Le nombre maximum de chemins arêtes-disjoints étant borné par le nombre d'arêtes, nous avons donc au plus $O(|V| \cdot |E|)$ itérations. L'algorithme est donc en $O(|V| \cdot |E|^2)$.

Algorithme distribué

Nous présentons maintenant l'algorithme distribué utilisé dans notre protocole de communication sûre des réseaux de capteurs. Les capteurs disposant de peu de ressources et n'ayant pas une connaissance globale du réseau, il est nécessaire de mettre au point une version distribuée de l'algorithme 8 décrit dans la section précédente. Chaque capteur v exécute un algorithme localement et communique avec les autres capteurs par des signaux (ou messages). En fonction des signaux reçus, le capteur v change d'état. Nous supposons que le processus de chaque capteur traite

l'information indépendamment de la réception des messages. En d'autres termes les capteurs stockent les signaux entrants dans une pile lorsqu'ils exécutent des étapes du processus. Nous ne prenons pas en compte les problèmes d'interférences dus à l'arrivée simultanée de plusieurs signaux ou les problèmes de collisions entre l'émission et la réception d'un signal.

Dans la version distribuée de l'algorithme, nous associons à chaque sommet v un numéro unique permettant de l'identifier dans le réseau. Par ailleurs, la variable θ_v représente l'appartenance d'un sommet à un cluster. Si v appartient au cluster X_t , nous notons alors $\theta_v := t$, si v n'appartient à aucun cluster alors $\theta_v = NULL$. La variable d_v correspond à la distance courante en nombre de bonds entre v et θ_v , $d_v = \infty$ si $\theta_v = NULL$. Chaque sommet v dispose de la liste $\mathcal{N}_v$ de ses voisins dans le graphe $G'[X_{\theta_v}]$ ordonnée par ordre croissant en fonction de leur distance à t . Si $v \in T$, $d_v = 0$ et sinon $d_v = \min(d_u : u \in \mathcal{N}_v) + 1$. L'ajout d'un sommet u à $\mathcal{N}_v$ implique la mise à jour de la distance d_v . Nous définissons pour chaque capteur v une table de routage local $\mathcal{R}_v$ qui indique pour chaque sommet u le sommet auquel un message arrivant en v provenant de u doit être transmis. Par exemple si nous utilisons un chemin $p = \{\dots, u, v, w, \dots\} \in \mathcal{P}$ nous aurons $\mathcal{R}_v(u) = w$ et $\mathcal{R}_v(w) = u$. Si $\mathcal{R}_v(u) = NULL$, alors l'arête vu n'est affectée à aucun chemin et appartient encore à G' . De plus, chaque capteur v dispose d'une liste de demi-chemin $\mathcal{T}_v$. Un demi-chemin est un couple (capteur, terminal). Celui-ci intervient lors de la recherche des chemins. L'algorithme étant distribué, lorsqu'un capteur demande la recherche de deux demi-chemins pour construire un route, il recevra les réponses de manières différées. La variable $\mathcal{T}_v$ permet au capteur v de stocker un demi-chemin en attendant l'autre moitié. Les demi-chemins sont gardés un certain temps en mémoire puis supprimés, pour éviter de garder inutilement un demi-chemin inexploité trop longtemps. Si un capteur n'a plus de signaux à traiter, il libère un demi-chemin au bout d'une durée pré-définie. Libérer un demi-chemin signifie que nous demandons, par bonds successifs, aux capteurs du demi-chemin de mettre à jour leur table de routage local.

Les signaux utilisés dans notre algorithme sont décrits dans le tableau 5.1 suivant. Le paramètre m de chaque signal est l'identifiant réseau du capteur émetteur.

Un signal peut être envoyé à tous les capteurs ou à un capteur en particulier qui dans ce cas est spécifié dans l'algorithme. Nous supposons que chaque terminal t émet les signaux $csc(m, t, d)$, $succes(m, t)$ de manière cohérente, i.e. chaque terminal t initie l'émission du message $csc(t, t, 0)$ et pour chaque réception de message $recherche(m, t)$ il répond $succes(t, t)$.

	Signal	Définition
(1)	$csc(m, t, d)$	Le capteur t souhaite communiquer. La variable d est la distance courante entre m et t .
(2)	$recherche(m, t)$	m demande la recherche d'un chemin vers t via m .
(3)	$succes(m, t)$	Succès de la recherche d'un chemin vers t via m .
(4)	$echec(m, t)$	Échec de la recherche de chemin vers t via m .
(5)	$libre(m)$	Le capteur m est libre, il appartient à aucun cluster.

TABLE 5.1 – Signaux pour l'algorithme distribué

La figure 5.7 représente une vision globale des transitions entre les deux états d'un capteur en fonction des signaux reçus. Les signaux sont représentés par leur numéro dans le tableau 5.1.

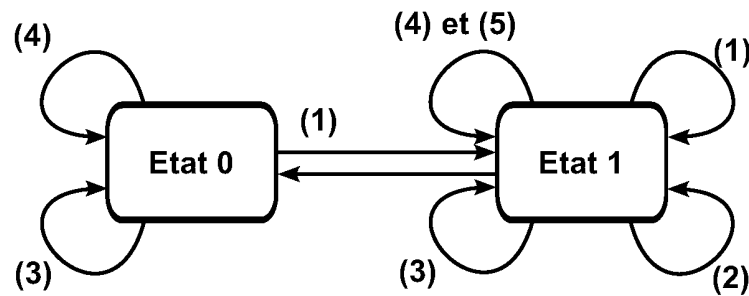


FIGURE 5.7 – Transition des états en fonction des signaux

Nous présentons maintenant une description formelle de l'algorithme distribué pour déterminer un routage des communications tolérant les pannes dans les réseaux critiques de capteurs.

Algorithme 9: Algorithme distribué de routage sûr dans les réseaux critiques de capteurs

Données : Réception d'un signal sig

```

// Le capteur dispose d'une variable Etat initialisée à 0;
1 si Etat = 0 alors
2   | Capteur-libre(sig);
3 fin
4 sinon
5   | Capteur-actif(sig);
6 fin

```

L'algorithme 9 fait appel à deux procédures qui correspondent aux deux états

possibles du capteur.

La procédure *Capteur-libre* représente l'état 0 du capteur v . Elle décrit le comportement d'un capteur v qui n'appartient à aucun cluster. Sa principale fonction est d'associer le capteur v à un cluster, pour cela il écoute ses voisins. Si un de ses voisins appartient à un cluster alors v considère qu'il appartient lui aussi à ce cluster et passe à l'état 1 (ligne 1 à 6), à la réception d'un signal $csc(m, t, d)$, le capteur v affecte sa variable θ_v à t . Cela signifie que v appartient au cluster X_t . Ensuite il met à jour sa liste de voisins courants (fonction *Maj-Voisins*(m, d) à la ligne 3). Cette fonction a pour but de mettre à jour la liste des voisins courants $\mathcal{N}_v$ utilisée par la suite par v pour la recherche de chemin jusqu'à t . Ensuite le capteur v informe tous ses voisins (dans le graphe G) qu'il appartient au cluster X_t (ligne 4). Pour finir, v passe à l'état 1. Pour sortir de l'état 0 un capteur doit recevoir un signal $csc(m, t, d)$, mais v peut réceptionner d'autres signaux. La réception d'un signal *libre*(m) est ignoré. En effet, le signal *libre*(m) indique que le capteur m voisin de v n'appartient à aucun cluster, v étant dans le même état que ce dernier, il est inutile de lui répondre. Les signaux *recherche*(m, t), *succes*(m, t) et *echec*(m, t) peuvent être reçus même si v est dans l'état 0. Cela se produit lorsque v a appartenu au cluster de t et que son voisin m le considère encore comme tel (il n'a pas encore traité le message *libre*(v)). Si le signal *recherche*(m, t) est reçu, v répond qu'il ne peut pas atteindre t par le message *echec*(v, t) (ligne 7 à 9). Le capteur v n'étant dans aucun cluster la recherche de chemin échoue. Le signal *succes*(m, t) est une réponse positive à une recherche de chemin vers t via m (ligne 10 à 12). Cela signifie que dans le passé v a demandé à m de rechercher un chemin vers t et que maintenant v reçoit la confirmation qu'un chemin entre v et t via m est disponible dans le graphe. Le capteur v appelle la fonction *MajP*(m, t) qui en fonction des cas : retransmet le message de succès si v avait été sollicité (dans le passé) par un de ses voisins pour effectuer une recherche vers t , et sinon (v avait initié la recherche du demi-chemin) il construit une route entre deux terminaux ou conserve le demi-chemin en attendant la réponse du second demi-chemin permettant de construire une route entre deux terminaux par concaténation. Le signal *echec*(m, t) est une réponse négative à une recherche de chemin vers t via m effectuée dans le passé (ligne 13 à 20). Si v avait initié la demande alors il n'existe pas de capteur à qui router une transmission de m dans la table de routage locale $\mathcal{R}_v$. Dans ce cas rien ne doit être fait car v ne connaît à cet instant aucun voisin appartenant au cluster X_t pouvant être sollicité pour une recherche de chemin vers t . Si v n'avait pas initié la recherche du chemin (ligne 14), alors v indique à n , le voisin ayant demandé la recherche du chemin, qu'il

ne peut pas trouver de chemin vert t . Pour finir v rajoute à G' les arêtes mv et vn (ligne 17 et 18).

Procédure *Capteur-libre*(sig)

```

1 si sig = csc( $m, t, d$ ) alors
2   |  $\theta_v := t$  ;
3   | Maj-Voisins( $m, d$ ) ;
4   | Envoyer à tous les voisins dans  $G$  le signal csc( $v, t, d_v$ ) ;
5   | Etat := 1 ;
6 fin
7 sinon si sig = recherche( $m, t$ ) alors
8   | Envoyer à  $m$  le signal echec( $v, t$ ) ;
9 fin
10 sinon si sig = succes( $m, t$ ) alors
11   | MajP( $m, t$ );
12 fin
13 sinon si sig = echec( $m, t$ ) alors
14   | si  $\mathcal{R}_v(m) \neq NULL$  alors
15     |  $n := \mathcal{R}_v(m)$  ;
16     | Envoyer à  $n$  le signal echec( $v, t$ ) ;
17     |  $\mathcal{R}_v(m) := NULL$  ;
18     |  $\mathcal{R}_v(n) := NULL$  ;
19   | fin
20 fin

```

La procédure *Capteur-libre* fait appel à des sous-procédures décrites ci-dessous. La fonction *Maj-Voisins*(m, d) met à jour la liste de voisins courants $\mathcal{N}_v$ utilisée par la suite par v pour la recherche de chemins. Puis, v effectue la mise à jour de la distance entre le capteur v et son terminal. Le test de la ligne 1 vérifie si l'arête mv appartient à G' .

Fonction *Maj-Voisins*(m, d)

```

1 si  $\mathcal{R}_v(m) = NULL$  alors
2   | Ajouter  $m$  à  $\mathcal{N}_v$  pour une distance  $d$  ;
3   |  $d_v := \min(d_u : u \in \mathcal{N}_v) + 1$  ;
4 fin

```

5.2. ROUTAGE SÛR DE L'INFORMATION DANS LES RÉSEAUX DE CAPTEURS

La fonction $MajP(m, t)$ traite une réponse positive de recherche de chemin. Si le capteur n'a pas initié la recherche du demi-chemin, il se contente de transmettre l'information au capteur qui lui en a fait la demande (ligne 1 à 4). En effet, si $\mathcal{R}_v(m) \neq NULL$ alors il existe $n = \mathcal{R}_v(m)$ voisin de v qui a (dans le passé) demandé à v de rechercher un chemin vers t . Si le capteur a initié la recherche du demi-chemin, alors il vérifie s'il peut créer un chemin complet avec les demi-chemins déjà en sa possession, sinon il le stocke en attendant l'arrivée d'un autre demi-chemin (ligne 5 à 14). La création du routage s'effectue si v a déjà dans sa liste de demi-chemins $\mathcal{T}_v$, un demi-chemin vers un autre terminal $t' \neq t$ (ligne 6). Dans ce cas, il supprime les arêtes uv et vm de G' (ligne 7 et 8), ainsi que le demi-chemin vers t' (ligne 9). Dans le cas où aucun demi-chemin vers $t' \neq t$ existe (ligne 11) v stocke le demi-chemin vers t via m dans $\mathcal{T}_v$ (ligne 12).

Fonction $MajP(m, t)$

```

1 si  $\mathcal{R}_v(m) \neq NULL$  alors
2   |    $n = \mathcal{R}_v(m)$  ;
3   |   Envoyer à  $n$  le signal succes( $v$ ) ;
4 fin
5 sinon
6   |   si  $\exists(u, t') \in \mathcal{T}_v$  alors
7   |   |    $\mathcal{R}_v(u) := m$  ;
8   |   |    $\mathcal{R}_v(m) := u$  ;
9   |   |   Supprimer  $(u, t')$  de  $\mathcal{T}_v$  ;
10  |   fin
11  |   sinon
12  |   |   Ajouter  $(m, t)$  à  $\mathcal{T}_v$ ;
13  |   fin
14 fin

```

La procédure *Capteur-actif* représente l'état 1 d'un capteur v . Cette étape a pour but de faire grossir le cluster X_t (ligne 12 à 14) et de détecter une liaison entre deux clusters pour essayer d'établir une connexion entre les terminaux de chaque cluster (ligne 1 à 11). Lorsque v reçoit le signal $csc(m, t, d)$ d'un voisin m appartenant à un cluster, il identifie si m appartient au même cluster que lui (ligne 2). Si cela est le cas, v met à jour sa liste de voisins courants $\mathcal{N}_v$ via la fonction *Maj-Voisins*(m, d) (ligne 8 à 10). Sinon v a détecté que l'arête $vm \in E(G')$ est incidente aux deux clusters X_t et X_{θ_v} (ligne 2 à 7). Le capteur v essaie alors de trouver deux demi-chemins. Le premier entre lui-même et le terminal θ_v via la fonction *TrouverT*(m, θ_v) (ligne 3). Cette fonction envoie un message *recherche*(n, θ_v) au voisin courant $n \in \mathcal{N}_v$ et supprime l'arête vn de G' . Si aucun voisin courant n'est disponible alors v détecte qu'il n'appartient plus au cluster X_{θ_v} et passe à l'état 0. Si v reste à l'état 1 après la recherche du demi-chemin de v à θ_v (ligne 4), il lance une recherche du demi-chemin entre lui-même et t (le terminal du cluster de m). Cette recherche se traduit par l'envoi à m du signal *recherche*(v, t) (ligne 5). Cette état gère aussi la réception des messages relatifs à la recherche des demi-chemins. Si le signal *recherche*(m, t) est reçu (ligne 15) et que le terminal t recherché est le terminal du cluster de v (ligne 16), le capteur v effectue une recherche vers t via un de ses voisins courants $n \in \mathcal{N}_v$ via la fonction *TrouverT*(m, t) (ligne 17). Si t n'est pas le terminal du cluster de v , le capteur répond qu'il ne peut pas atteindre t (ligne 19 à 21). Le signal *succes*(m, t) est identique à l'état 0 (ligne 23 à 25). Si le signal *echec*(m, t) est reçu (ligne 26) et que le terminal t recherché est le terminal du cluster de v (ligne 27), le capteur v effectue une recherche vers t (ligne 28). Si t n'est pas le terminal du cluster de v et que v n'a pas initié la recherche du demi-chemin, c'est à dire qu'il existe un sommet n pour router les communications en provenance de m dans $\mathcal{R}_v$ (ligne 30) ; le capteur v envoie à n un signal d'échec de recherche de chemin et rajoute l'arête vn à G' (ligne 32). Dans tous les cas, la réception d'un signal d'échec émit par m entraîne le rajout de l'arête mn à G' (ligne 34).

```

Procédure Capteur-actif(sig)


---


1 si  $sig = csc(m, t, d)$  alors
2   si  $\theta_v \neq t$  alors
3      $Etat := TrouverT(m, \theta_v);$ 
4     si  $Etat = 1$  alors
5       Envoyer à  $m$  le signal  $recherche(v, t)$  ;
6     fin
7   fin
8   sinon
9      $Maj-Voisins(m, d)$  ;
10  fin
11 fin
12 sinon si  $sig = libre(m)$  alors
13   Envoyer à  $m$  le signal  $csc(v, \theta_v, d_v)$  ;
14 fin
15 sinon si  $sig = recherche(m, t)$  alors
16   si  $\theta_v = t$  alors
17      $Etat := TrouverT(m, \theta_v);$ 
18   fin
19   sinon
20     Envoyer à  $m$  le signal  $echec(v, t)$  ;
21   fin
22 fin
23 sinon si  $sig = succes(m, t)$  alors
24    $MajP(m, t);$ 
25 fin
26 sinon si  $sig = echec(m, t)$  alors
27   si  $\theta_v = t$  alors
28      $Etat := TrouverT(m, t);$ 
29   fin
30   sinon si  $\mathcal{R}_v(m) \neq NULL$  alors
31     Envoyer à  $n := \mathcal{R}_v(m)$  le signal  $echec(v, t)$  ;
32      $\mathcal{R}_v(n) := NULL$  ;
33   fin
34    $\mathcal{R}_v(m) := NULL$  ;
35 fin

```

La procédure *Capteur-actif* fait appel à une sous-procédure décrite ci-dessous. La procédure *TrouverT*(m, t) recherche un chemin de v à t . Elle interroge les voisins courants de la table des voisins dans l'ordre (ligne 1 à 7). Pour cela, v supprime n le sommet le plus proche "localement" de t de ses voisins courants (ligne 2), retire les arêtes vn et mv de G' , envoie à n une demande de recherche de chemin vers t (ligne 4 et 5) et reste à l'état 1 (ligne 6). Si la table des voisins est vide, alors le capteur v comprend qu'il n'appartient plus au cluster X_{θ_v} (ligne 8 à 13). Dans ce cas, l'arête vm est rajoutée à G' (ligne 9), v informe m de l'échec de la recherche du demi-chemin (ligne 10), signale à tous ses voisins qu'il est libre (ligne 11) et passe à l'état 0 (ligne 12).

Fonction *TrouverT*(m, t)

```

1 si  $\exists n \in \mathcal{N}_v$  tel que  $\mathcal{R}_v(n) = NULL$  et  $d_n \leq d_x \forall x \in \mathcal{N}_v$  alors
2   Supprimer  $n$  de  $\mathcal{N}_v$  ;
3    $\mathcal{R}_v(n) := m$  ;
4    $\mathcal{R}_v(m) := n$  ;
5   Envoyer à  $n$  le signal recherche( $v, t$ ) ;
6   Renvoyer la valeur 1;
7 fin
8 sinon
9    $\mathcal{R}_v(m) = NULL$  ;
10  Envoyer à  $m$  le signal echec( $v$ ) ;
11  Envoyer à tous les voisins le signal libre( $v$ ) ;
12  Renvoyer la valeur 0;
13 fin

```

Analyse de l'algorithme distribué 9

Comme pour l'algorithme centralisé 8, l'analyse de l'algorithme distribué 9 se fait en trois phases. La première phase consiste à faire grossir les clusters. Les deux états d'un capteur participent à cette phase. Pour un capteur v en état 0, la présence d'un voisin appartenant à un cluster X_t implique l'appartenance de v à X_t (ligne 1 à 6). Pour un capteur v en état 1, un voisin n qui sort d'un cluster (réception d'un signal $libre(n)$) est informé par v qu'il peut faire partie du même cluster que lui (ligne 12 à 14). La deuxième phase consiste à identifier les paires de sommets voisins appartenant à deux clusters différents. Pour cela, un capteur v dans l'état 1 appartenant à X_t détecte un voisin n appartenant à un cluster différent $X_{t'}$ grâce à la réception d'un messages ($csc(n, t', d)$) de la phase précédente (ligne 1 à 10). Dans ce cas, v essaie de construire deux demi-chemins pour relier t à t' en passant par l'arête vn (ligne 3 et 5). La troisième phase consiste à rechercher les demi-chemins. Pour garantir des chemins disjoints par les arêtes, les arêtes parcourus lors de la recherche du chemin sont supprimées de G' . En effet, lorsqu'un capteur v interroge un voisin courant $n \in \mathcal{N}$, il supprime l'arête vn de G' en attendant la réponse (ligne 3 de la fonction $TrouverT(m, t)$). Si la recherche du chemin via n est un échec l'arête nv est rajoutée à G' . Cela se traduit du côté de v par la ligne 34 du processus *Capteur-actif* et du côté de n à ligne 9 de la fonction $TrouverT(m, t)$. Lors de la recherche les capteurs sollicités détectent s'ils appartiennent encore à leur cluster. Si cela n'est plus le cas, le capteur passe à l'état 0 (ligne 8 à 13 de la fonction $TrouverT(m, t)$).

Cet algorithme distribué souffre du fait que les clusters ne grossissent pas d'une manière uniforme comme dans l'algorithme centralisé. De plus la recherche des chemins est concurrente. Ceci peut entraîner des chemins plus longs et des échecs qui ne seraient pas apparus dans la version centralisée. Par exemple avec des arêtes réservées par un demi-chemin qui ne seront pas utilisées dans le routage final car le deuxième demi-chemin associé n'existe pas.

Résultats des simulations

Nous présentons maintenant succinctement les simulations effectuées. Nous avons implémenté quatre algorithmes. Les deux premiers sont ceux présentés précédemment, c'est-à-dire l'algorithme 8 nommé *Algo_Centralise* et sa version distribuée nommée *Algo_Distribue*. Le troisième algorithme, nommé *Algo_Optimal*, est un algorithme polynomial [102] qui résout le problème de Mader. Ce dernier calcule des solutions optimales qui permettent d'évaluer expérimentalement la qualité des solutions obtenues avec les heuristiques. Enfin le quatrième algorithme, nommé *Algo_PCC* consiste à rechercher p le plus court chemin entre toutes les paires de terminaux, d'ajouter p à $\mathcal{P}$ et de retirer du graphe les arêtes de p et à recommencer tant qu'il existe un chemin entre deux terminaux.

La génération aléatoire des instances du problème prend trois paramètres : le nombre de sommets, le nombre d'arêtes et le nombre de terminaux. Le tableau 5.2 décrit les résultats obtenus.

Chaque ligne nommée *NbSommets_NbAretes_NbTerminaux* est la moyenne des résultats obtenus sur 100 instances. Pour une instance I , nous définissons $Val1(I)$ le nombre de chemins renvoyés par *Algo_Centralise* sur l'instance I , de manière similaire $Val2(I)$ pour *Algo_Distribue* et $opt(I)$ le nombre maximum de chemins dis-joints (renvoyé par *Algo_Optimal*). Nous définissons quatre critères de performance. Pour une instance I , $Centralise = \frac{opt(I) - Val1(I)}{opt(I)}$, $Distribue$ qui est équivalent à $Centralise$ pour *Algo_Distribue* et Nb_Mess qui correspond au nombre moyen de signaux envoyés par un sommet du graphe pour l'algorithme distribué. Enfin PCC est l'équivalent de $Centralise$ et $Distribue$ pour l'algorithme *Algo_PCC*.

5.2. ROUTAGE SÛR DE L'INFORMATION DANS LES RÉSEAUX DE CAPTEURS

instance	<i>Centralise</i>	<i>Distribue</i>	$\frac{Nb_Mess}{Nb_Arêtes}$	<i>PCC</i>
25S_50A_10T	2 %	49.1 %	9.9	0.9 %
30S_60A_10T	1.9 %	58.4 %	12.1	0.7 %
40S_80A_10T	1.5 %	69.7 %	15.2	0.4 %
50S_100A_10T	0.7 %	66.4 %	18.8	0.6 %
50S_100A_15T	1.4 %	61.4 %	20.5	0.1 %
50S_100A_20T	1 %	47.6 %	21.9	0.1 %
100S_200A_20T	0.7 %	61 %	19.7	0.2 %
100S_200A_30T	1.1 %	58.9 %	21.6	0.1 %
100S_200A_40T	1.2 %	46.8 %	23.3	0.04 %
100S_200A_50T	1.5 %	35 %	23.6	0 %
100S_300A_20T	0.9 %	63.3 %	21	0.5 %
100S_300A_30T	2.1 %	56.5 %	23.8	1 %
100S_300A_40T	0.8 %	45.2 %	34.6	0.2 %
100S_300A_50T	0.6 %	35.8 %	34.6	0.1 %
100S_400A_20T	1 %	69 %	38.3	0.2 %
100S_400A_30T	0.9 %	57.4 %	44.9	0.1 %
100S_400A_40T	0.8 %	42.8 %	47.9	0.3 %
100S_400A_50T	1.7 %	36.4 %	47.8	0.6 %

TABLE 5.2 – Résultats expérimentaux du protocole de routage

Nous constatons premièrement que l'algorithme *PCC* donne de très bonnes solutions. Cette heuristique naturelle essaie d'économiser les arêtes utilisées pour router une communication entre deux terminaux. L'algorithme *Centralise* donne lui aussi de très bonnes solutions car il s'inspire de *PCC* en faisant grossir des ensembles. Si on réinitialise les ensembles à leur seul terminal après chaque ajout de chemin, nous pouvons les considérer comme des boules. L'algorithme serait alors équivalent à *PCC* faisant grossir des boules et, dès que deux boules se touchent (une arête appartient à deux boules), la paire de terminaux la plus proche est détectée. Enfin, l'algorithme *Distribue* souffre des contraintes techniques imposées. Plus particulièrement, la recherche des chemins s'effectue de manière concurrente dans un contexte asynchrone dégrade la qualité de la solution en produisant des chemins de routage plus longs. Pour finir, bien qu'aucune garantie théorique ne soit apportée, nous constatons que le nombre de messages échangés dépend principalement du nombre d'arêtes du graphe. Ceci est concordant avec un algorithme basé sur la recherche de chemin. Le nombre de messages échangés est un critère d'évaluation important pour ce type d'algorithme.

5.3 Gestion de l'énergie dans les réseaux de capteurs

Dans cette section nous nous plaçons dans un réseau de capteurs identiques. Les capteurs sont supposés statiques et la topologie fixe. Les capteurs peuvent faire office de répéteur en transmettant les données sans produire d'information.

Pour acheminer de l'information dans ce type de réseau, il faut mettre en œuvre des protocoles de routage. Les réseaux de capteurs se distinguent actuellement des autres types de réseaux par la plus grande diversité de protocoles de routage. Les protocoles de routage les plus utilisés, peuvent être regroupés dans quatre principales familles. Les protocoles de routage par inondation se contentent de propager l'information d'un capteur à tous ses voisins, qui agissent de façon identique, et ce jusqu'à atteindre la destination. L'inconvénient majeur de la multiplication des échanges est la consommation d'énergie, qui est souvent une ressource précieuse dans les réseaux de capteurs. Les protocoles où les routes sont établies en fonction des besoins. Ces protocoles ont l'avantage d'adapter le routage à l'état du réseau mais ont l'inconvénient de demander un calcul de route fréquent qui nécessite lui aussi beaucoup d'échanges radio. Les protocoles qui utilisent des tables de routage qui sont pré-calculées ou établies par les éléments constitutifs du réseau. Citons parmi ces derniers, le protocole Fisheye State Routing et Optimized Link State Routing. Lorsque le réseau comporte un grand nombre d'éléments, il devient indispensable de concevoir des réseaux hiérarchiques pour réduire les tables de routages. Les protocoles de routage par la source, dans lesquels les routes sont prédéterminées et dans lesquels le capteur qui émet un paquet indique dans l'entête du paquet la liste des capteurs ou répéteurs qui constituent le chemin de routage jusqu'à sa destination. C'est dans cette dernière famille de protocole que se positionne notre travail.

Les enjeux de la gestion du routage

Les éléments des réseaux de capteurs sont en général alimentés par batterie. Dans le cadre de cette étude nous supposons que cette batterie, ne peut être ni rechargée ni changée. La durée de vie limitée de la batterie détermine la durée de vie du capteur et par la même celle du réseau. Il est donc important de réduire le plus possible la consommation d'énergie des capteurs. La consommation d'énergie d'un capteur dépend des trois fonctions suivantes. La transmission des données est la fonction la plus vorace en énergie. Elle représente en moyenne la moitié de la consommation électrique. Elle est dépendante de la distance et de l'environnement

qui sépare l'émetteur du récepteur. La réception des paquets émis par d'autres capteurs ou répéteurs. Cette fonction est moins consommatrice que la transmission mais représente le deuxième poste de consommation. Pour finir, le prélèvement des données, c'est-à-dire la mesure de l'environnement, a une consommation d'énergie qui dépend grandement du type de mesure à effectuer.

Pour économiser l'énergie dispersée lors de la transmission et de la réception la solution la plus couramment consiste à mettre en sommeil les équipements [6, 68, 127]. Nous considérons alors que les capteurs sont *passifs*.

Nous considérons que la connexité du réseau est perdue dès qu'il n'est plus possible d'obtenir une mesure requise par le système. La fonction la plus consommatrice d'énergie étant la transmission, il est certain que les composants qui tomberont en panne les premiers sont ceux qui sont le plus sollicités de ce point de vue. La manière dont est routée l'information a donc un impact sur la durée de vie du réseau. C'est pour cette raison que nous décrivons un protocole de routage qui, en fonction de la consommation d'énergie et de l'état de chaque composant, permet d'obtenir la plus grande durée de vie du réseau. Des processus tenant compte de la durée de vie du réseau sont décrits dans [33] et [58], mais les auteurs prennent le parti de minimiser l'énergie installée dans le réseau pour qu'il puisse vivre jusqu'à un temps donné. De plus il s'intéresse plus à l'installation des répéteurs qu'au routage des données. D'une manière générale ces méthodes sont décentralisées et ainsi chaque composant du réseau détermine vers quel autre composant il va émettre de sorte que ses propres émissions le conduisent à utiliser le moins d'énergie possible. Le critère d'optimisation, quand il existe, est la minimisation de la consommation totale d'énergie, cela ne garantit pas que cette consommation ne sera pas supportée par un petit nombre de composants, entraînant de ce fait une panne plus rapide de ceux-ci. C'est pourquoi nous proposons ci-après un processus centralisé ayant une connaissance complète du réseau et recherchant la plus grande durée de vie du réseau en tenant compte de l'énergie disponible de chaque composant.

Protocole de routages

Une station de base qui souhaite interroger un capteur doit effectuer une communication par bonds en calculant le routage. Le message ainsi produit contient la totalité du chemin à emprunter. Ainsi chaque capteur intermédiaire du chemin sait à qui il doit transmettre l'information. Le routage permet donc de transmettre un signal au capteur lui demandant d'effectuer une mesure et permet aussi à ce capteur d'envoyer la donnée jusqu'à la station de base. Même si ce n'est pas toujours le cas,

nous considérons que le message retour emprunte la même route que le message aller.

Notre problème consiste donc à trouver un ensemble de routes pour transmettre l'information. Nous considérons la durée de vie du réseau comme critère d'évaluation de ce problème d'optimisation.

Les réseaux pour lesquels nous cherchons à déterminer des routes sont des réseaux constitués d'un ensemble de capteurs V_c et d'un point de collecte noté Ap . Le réseau est représenté par un digraphe $G = (V, E)$ où $V = \{Ap\} \cup V_c$ et E est l'ensemble des paires de sommets $\{i, j\}$ tel que le sommet i peut émettre physiquement en direction du sommet j et réciproquement.

Nous considérons un ensemble de périodes $T = \{t_1, \dots, t_n\}$ ordonnées chronologiquement. Nous supposons que nous pouvons planifier pour chaque capteur $c \in V_c$ les périodes de temps à partir desquelles un résultat de la mesure de son environnement est souhaité. Par exemple la température de l'eau, qui est consultée de manière régulière dans la journée pour informer les touristes, est inutile la nuit. Nous disposons donc, pour chaque capteur $c \in V_c$ et chaque période $t \in T$ d'une valeur binaire d_c^t qui est égale à 1 si une demande d'information est requise à partir du temps t et à 0 sinon. Nous ne prenons pas en compte le temps de transit de la demande dans le réseau et la considérons comme instantanée.

L'énergie disponible en un point $c \in V_c$ du réseau est notée ε_c . L'énergie consommée par le capteur c pour produire les données d'un paquet est notée $\pi^p(c)$. Nous notons $\pi^e(i, j)$ l'énergie consommée pour transmettre un paquet de i vers j et $\pi^r(i)$ pour recevoir un paquet en i .

Chaque chemin est considéré comme valide si l'énergie disponible dans les capteurs qu'il traverse est suffisante pour transmettre l'information. Une demande de capteur est satisfaite s'il existe un chemin entre Ap et ce capteur. Nous considérons le réseau mort lorsque qu'il n'existe plus de chemin pour satisfaire une demande d'un capteur.

Le problème que nous considérons peut être formulé de la façon suivante :

Problème: Routage Durée de vie Max

Entrée: Un digraphe $G = (V, E)$, un ensemble de capteurs $V_c \subset V$, un ensemble de périodes $T = \{t_0, \dots, t_n\}$, pour chaque capteur $c \in V_c$ et chaque période de temps demandées $d_c^t \in \{0, 1\}$, un point de collecte $Ap \in V$

Question: Quelle est la plus grande valeur de i tel qu'il existe un routage des communications qui permettent de transmettre toutes les informations requises jusqu'au temps t_i ?

Soit $\mathcal{P}_c$ l'ensemble des chemins reliant Ap à un sommet c et $\mathcal{P} = \bigcup_{c \in V_c} \mathcal{P}_c$ l'ensemble des chemins du graphe.

Nous décrivons le programme linéaire en nombres entiers $R2$ qui nous permet de déterminer les routages qui conduisent à la plus grande longévité du réseau. Nous utilisons les variables suivantes. Pour $t \in T$, la variable y^t vaut 1 si le réseau est en état de marche à l'instant t et 0 sinon. Pour tout $t \in T$ et $p \in \mathcal{P}$, la variable x_p^t vaut 1 si p est utilisé durant la période t et que le réseau est en état de marche ; 0 sinon.

$$R2 = \left\{ \begin{array}{ll} \begin{array}{l} \text{Max} \quad \sum_{t \in T} y^t \\ \text{s.c} \end{array} & \\ y^{t_i} - y^{t_{i-1}} \leq 0, \quad \forall t_i \in T - t_0 & (1) \\ y^t - \sum_{p \in \mathcal{P}_c} x_p^t \leq 0, \quad \forall c \in V_c, \quad \forall t \in T \text{ tel que } d_c^t = 1 & (2) \\ \sum_{\substack{p \in \mathcal{P} \\ p \ni (c,v) \\ \text{ou} \\ p \ni (v,c)}} (\pi_c^e(v) + \pi_c^r) x_p^t + \sum_{p \in \mathcal{P}_c} \pi_c^{pr} x_p^t \leq \varepsilon_c, \quad \forall c \in V_c, \quad \forall t \in T & (3) \\ x_p^t \in \{0, 1\}, \quad \forall t \in T, \quad \forall p \in \mathcal{P} & \\ y^t \in \{0, 1\}, \quad \forall t \in T & \end{array} \right.$$

La fonction objectif consiste à maximiser la durée de vie du réseau. Les contraintes (1) interdisent de ressusciter le réseau, c'est à dire qu'un réseau mort reste mort

jusqu'à la dernière période de temps. Les contraintes (2) définissent un réseau vivant à l'instant t si pour tout capteur $c \in V_c$ devant effectuer une mesure à cet instant ($d_c^t = 1$) il existe au moins une route permettant de faire transiter l'information. Les contraintes (3) expriment la consommation d'énergie de chaque capteur. Nous considérons que le message retour emprunte la même route que le message aller. Un chemin $p \in \mathcal{P}$ caractérise seulement un chemin aller, la consommation du chemin passant par un sommet c comprend naturellement l'arc aller cv_1 quand c est l'origine de l'arc. Mais la consommation de c prend aussi compte l'arc retour v_c , quand c est la destination, car c devra émettre vers v_2 pour le message retour.

Le programme linéaire en nombres entiers $R2$ contient un grand nombre de variables binaires.

Nous présentons maintenant la version de décision du problème précédent. Si nous fixons le temps de vie du réseau, le problème revient à satisfaire pour chaque capteur un nombre fixe de demandes. Nous notons $T_c(t) = \sum_{t_0}^t d_c^{t_0}$ le nombre de demandes à satisfaire pour le capteur $c \in V_c$ de t_0 à t .

Problème: Durée de vie Max

Entrée: Un digraphe $G = (V, E)$, un ensemble de capteurs $V_c \subset V$, un temps t , pour chaque capteur $c \in V_c$ e nombre de demandes à satisfaire $T_c(t)$, un point de collecte $Ap \in V$

Question: Existe-t-il un routage tel que le réseau reste vivant jusqu'au temps t ?

Nous présentons maintenant un algorithme permettant de résoudre le problème Routage Durée de vie Max. Cet algorithme utilise le programme linéaire en nombres entiers $R2(t)$ suivant. Les variables x_p correspondent cette fois-ci au nombre de fois où nous utilisons le chemin p pour satisfaire une demande. S'il existe un routage égale à la somme des $T_c(t)$ alors la réponse est oui, sinon la réponse est non.

$$R2(t) = \left\{ \begin{array}{l} \text{Max} \quad \sum_{p \in \mathcal{P}} x_p \\ \text{s.c} \quad \sum_{p \in \mathcal{P}_c} x_p \leq T_c(t), \quad \forall c \in V_c \\ \sum_{\substack{p \in \mathcal{P} \\ p \ni (c,v) \\ \text{ou} \\ p \ni (v,c)}} (\pi_c^e(v) + \pi_c^r) x_p + \sum_{p \in \mathcal{P}_c} \pi_c^{pr} x_p \leq \varepsilon_c, \quad \forall c \in V_c \quad (5) \\ x_p \in \mathbb{N}, \quad \forall p \in \mathcal{P} \end{array} \right. \quad (4)$$

La fonction objectif consiste à maximiser le nombre de chemins utilisés. Les contraintes (4) définissent un chemin pour chaque demande d'un capteur. Les contraintes (5) expriment la consommation d'énergie de chaque capteur.

L'avantage de la formulation $R2(t)$ est d'avoir des variables indépendantes des périodes de temps. Si la solution optimale renvoyée par $R2(t)$ est égale à $\sum_{c \in V_c} T_c(t)$, alors il est trivial de construire une solution réalisable de $R2$ de valeur t .

Proposition 5.3.1. *Le problème Durée de vie Max est NP-Complet.*

Démonstration 5.3.1.

Nous effectuons pour ce problème une réduction polynomiale à partir du problème NP-Complet Partition. Cette réduction se base sur la construction suivante.

Soit $X = \{s_1, \dots, s_n\}$ une instance du problème Partition et $\mathcal{S} = \sum_{s_i \in X} s_i$.

Nous définissons le digraphe G de la manière suivante. Les sommets $V(G) = (\{S, a, b, P\} \cup \{v_1, v_2, \dots, v_n\})$. Pour chaque sommet $v \in V$, l'énergie consommée lors de réception $\pi_v^r = 0$ et l'énergie consommée pour effectuer une mesure $\pi_c^{pr} = 0$. Pour les sommets a et b , nous avons la capacité énergétique de la batterie $\varepsilon_a = \varepsilon_b = \frac{\mathcal{S}}{2}$, l'énergie consommée pour émettre $\pi_a^e(P) = \pi_b^e(P) = 0$ et $\pi_P^e(a) = \pi_P^e(b) = 0$. De plus, $\varepsilon_S = \varepsilon_P = \infty$. L'ensemble des arcs inclut, pour tout $i \in \{1, \dots, n\}$ les arcs (v_i, a) , (v_i, b) et (S, v_i) . Pour tout $i \in \{1, \dots, n\}$ nous définissons $\varepsilon_{v_i} = 1$, $\pi_{v_i}^e(a) = \pi_{v_i}^e(b) = 1$, $\pi_a^e(v_i) = \pi_b^e(v_i) = s_i$, $\pi_S^e(v_i) = 0$ et $\pi_{v_i}^e(S) = 0$.

La figure 4.3 représente la construction précédente.

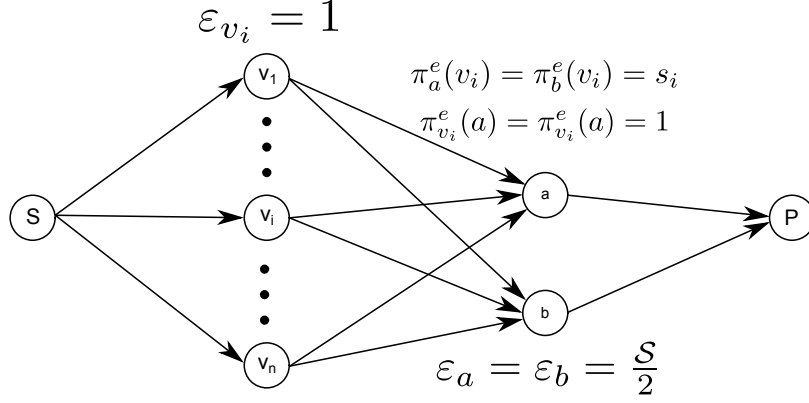


FIGURE 5.8 – Graphe d’une instance du problème Partition pour le problème de routage 2

Le problème $R2(t)$ consiste ici à réussir à router n demandes sur l’ensemble $\mathcal{P}$ des chemins reliant S à P pour une demande $T_P(t) = n$.

Pour commencer, nous montrons qu’une solution d’une instance X du problème Partition permet de construire une solution I de valeur n pour notre problème. Si la solution X admet une solution (P_a, P_b) alors à tous les éléments s_i appartenant à la partition P_a nous associons une route $\{S, v_i, a, P\}$. Nous effectuons la même construction pour la partition P_b . Cette construction respecte la capacité énergétique de chaque capteur. En effet, pour chaque v_i , il existe une unique transmission (v_i, a) ou (v_i, b) . De plus, la somme des transmissions des messages retour de a est égale à $\frac{\mathcal{S}}{2}$ car le routage est construit à partir d’une solution du problème Partition. La même observation est valable pour b . Nous avons donc construit une solution admissible du problème de routage.

Réciproquement une solution de l’instance I du problème de valeur n permet de construire une solution de l’instance X du problème de Partition. La solution d’une instance I n’excède pas la valeur n , car la demande de P pour un temps maximum de t est de n . Chaque sommet v_i l’énergie disponible ne lui permet de transmettre qu’à a ou b . De plus la solution de l’instance I est de valeur n donc tous les sommets v_i sont mis à contribution et transmettent soit à a soit à b . Il y a donc une partition des sommets entre a et b . L’énergie disponible pour émettre à partir de a est $\frac{\mathcal{S}}{2}$, par conséquent la somme des s_i pour les sommets v_i transmettant vers a est égale à $\frac{\mathcal{S}}{2}$. De manière similaire, à cause de la quantité d’énergie disponible en b , la somme des $s_{i'}$ des $v_{i'}$ transmettant vers b est égale à $\frac{\mathcal{S}}{2}$. Donc la somme totale des transmissions des v_i vers a et b est égale à $\mathcal{S}$. Donc nous avons bien une partition des sommets de

v_i entre a et b dont la somme des s_i associée est égale à $\frac{S}{2}$ pour chaque partition.

□

Pour trouver une solution au problème $R2$, nous effectuons une dichotomie pour trouver la plus grande valeur de t telle que le problème $R2(t)$ ait une solution satisfaisant toutes les demandes $T_c(t)$ des capteurs c . Sa résolution peut se faire par génération de colonne en utilisant la méthode de “branch and price”. Le sous-problème de séparation est un plus court chemin. Soit δ_c la variable duale associée à la contrainte (4) de $R2(t)$ et β_c la variable duale associée à la contrainte (5). Soit une solution optimale x^* sur un ensemble de chemin $\mathcal{P}'$ et δ^* et β_c^* la solution duale associée. Le sous-problème de séparation consiste à trouver un chemin $p \in \mathcal{P} - \mathcal{P}'$ tel que

$$\sum_{(c,v) \in p} \delta_c^* + \pi_c^e(v) \beta_c^* + \sum_{(v,c) \in p} \pi^r(c) \beta_c^* + \sum_{c: p \in P_c} \pi^{pr}(c) \beta_c^* < 1$$

Nous pouvons construire un digraphe $G^* = (V(G) \cup S, E(G))$ avec tous les arcs $(c, v) \in E(G)$ une longueur égale à $\pi_c^e(v) \beta_c^* + \pi^r(v) \beta_v^*$. Nous relions S à chaque $c \in V_c$ par un arc (S, v) de longueur $\pi^{pr}(c) \beta_c^* + \delta_c^*$. Le sous-problème de séparation est équivalent à trouver dans G^* un chemin entre S et Ap de longueur strictement inférieure à 1.

5.4 Conclusions

Dans ce chapitre nous traitons de deux problématiques liées aux réseaux capteurs et proposons deux protocoles de routage. Pour le premier problème, nous proposons un protocole de routage sûr dans les réseaux critiques de capteurs. Ce protocole prend en compte l’instabilité des communications sans fils dans ces types de réseaux. Il est basé sur un algorithme distribué qui calcule un ensemble maximum de chemins disjoints par les arêtes entre des capteurs souhaitant communiquer. Un brevet international a été déposé pour ce protocole [91]. Pour le second problème, nous proposons un protocole de routage qui maximise la durée de vie d’un réseau de capteurs. Nous prenons en compte la problématique liée à l’alimentation par batterie de la plupart des capteurs. Le protocole est centralisé et est basé sur des résolutions successives d’un programme linéaire en nombres entiers. Une preuve de NP-complétude est décrite pour ce problème se basant sur une réduction du problème Partition.

Les perspective envisageable pour ses deux problèmes sont les suivantes. Pour le premier protocole pour les réseaux de capteurs, il serait intéressant de proposer un algorithme pour des communications synchronisées. L'algorithme récemment proposé dans [88] calcule des plus courts chemins entre chaque paires de terminaux T d'un graphe en $O(|T| + D)$ où D correspond au diamètre du graphe. Il semble envisageable de l'adapter à ce problème, la synchronisation des communications pourrait s'inspirer des nombreuses méthodes efficaces déjà étudiées [8, 142, 155]. Une autre voie à explorer est de définir un routage dont le nombre de chemins disjoints par les arêtes est spécifié entre chaque paire de terminaux. Le problème correspond alors à un problème de multiflot entier difficile à résoudre et à approximer même dans le cas centralisé. La mise au point d'un algorithme distribué pour résoudre ce problème est donc une question importante mais difficile. Enfin, la prise en compte de la panne d'un capteur en imposant des routages disjoints par les sommets semble très naturelle. Là encore, le problème devient plus difficile à résoudre. En effet, il existe une réduction simple du problème du routage disjoint par les sommets en celui disjoint pas les arêtes. Pour cela il suffit d'ajouter un sommet sur chaque arête.

Pour le second protocole de routage, des simulations ainsi qu'une heuristique pour obtenir une solution entière sont des pistes à explorer. De manière équivalente à celle présenté dans le chapitre 3, l'heuristique d'arrondis pour une solution fractionnaire du programme linéaire $R2(t)$ consisterait par exemple pour chaque itération à fixer la valeur d'un chemin à sa partie entière. Nous obtenons alors un nouveau programme linéaire dont nous cherchons la solution optimale par génération de colonnes (branch and price). L'algorithme pourrait privilégier le chemin non fixé dont la valeur fractionnaire est la plus proche de sa partie entière (inférieure). De cette manière, il existe toujours une solution réalisable et le nombre d'itérations est de l'ordre du nombre de demandes totales des capteurs. Enfin, des évolutions du modèle en fonction des besoins des opérateurs peuvent être considérées. Par exemple, la prise en compte de messages descendants émis par les capteurs à intervalles réguliers. Dans ce cas, le routage sera probablement unique entre le capteur et la station de base. Une seconde évolution pourrait être de permettre un routage aller différent du routage retour. La première évolution est possible via une modification mineure du programme linéaire. La seconde évolution impose d'ajouter les chemins retour entre chaque capteur et la station de base à l'ensemble des chemins. Là encore une modification mineure du programme linéaire est à effectuer.

Dans ce travail de thèse, nous nous sommes intéressés à l'utilisation d'outils et de modèles développés dans le cadre de l'algorithmique des graphes pour la résolution de problèmes d'optimisations dans les réseaux de télécommunications.

Dans la première partie de notre travail, présentée dans le chapitre 3, nous proposons une modélisation unique regroupant les deux problématiques liées au déploiement de réseaux FTTH. Cette modélisation est basée sur la notion de flot généralisé avec contraintes de longueurs sur les chemins. Contrairement aux études précédentes qui traitent séparément le problème du positionnement des coupleurs optiques et celui du choix des fibres à déployer ou imposent des niveaux de coupleurs, nous décrivons un linéaire en nombres entiers qui modélise simultanément ces deux aspects du processus de conception. Nous mettons en place une méthode de génération de colonnes pour résoudre la relaxation linéaire de ce programme linéaire en nombres entiers. Le sous-problème de séparation correspond à un problème de plus court chemin généralisé avec contraintes de ressources. Nous résolvons ce sous-problème par un algorithme de programmation dynamique pseudo polynomial. Enfin, nous proposons une méthode d'arrondis pour obtenir une solution entière à partir de la solution fractionnaire. L'intérêt principal de ce travail est double. D'une part, le modèle proposé permet de regrouper la localisation des coupleurs optiques et le choix du câblage. D'autre part, la solution proposée va permettre d'évaluer de façon plus rigoureuse les règles d'ingénierie qui sont actuellement utilisées par l'opérateur pour le déploiement des fibres sur le terrain. Les simulations effectuées sur un ensemble d'instances artificielles de grandes tailles donnent de très bonnes solutions.

Dans le chapitre 4, nous étudions un problème d'optimisation liés à la qualité de service dans les réseaux de télécommunications. Ce modèle est couramment modélisé par un problème de flot avec contrainte de délai convexe. Nous présentons

une méthode qui permet d'obtenir une solution approchée du problème de flot avec contrainte de délai convexe à partir d'une solution optimale d'un problème de flot avec contrainte de délai linéaire. Nous étudions particulièrement le cas où la fonction de délai est proportionnelle et présentons deux preuves de NP-difficulté. Ces preuves se basent sur une réduction polynomiale à partir d'un problème Partition. Nous présentons ensuite un algorithme d'approximation de facteur 2 sur le graphe chemin avec demandes uniformes. Cet algorithme consiste à calculer un stable de poids maximum dans le graphe d'intervalles des demandes pour obtenir une solution optimale sans intersection. Les graphe d'intervalles étant parfait, il existe un algorithme polynomial pour y calculer le stable de poids maximum. Nous démontrons le facteur d'approximation en créant à partir de la solution primale réalisable une solution duale réalisable dont la valeur est au plus deux fois celle du primale. Enfin, nous proposons un programme linéaire à partir de la modélisation en programmation disjonctive de flot avec contrainte proportionnelle. Ce programme linéaire ayant un grand nombre de variables et de contraintes nous proposons une méthode de recherche locale pour le résoudre. Cette méthode utilise les informations obtenues en résolvant un programme linéaire réduit pour explorer l'espace des combinaisons des chemins. Les simulations effectuées sur des instances de grandes tailles générées aléatoirement montrent que cette approche permet d'obtenir de bons résultats en un temps raisonnable.

Le chapitre 5 porte sur deux problématiques liées aux réseaux de capteurs. Premièrement, nous proposons un protocole de routage pour les réseaux critiques. Ce protocole maximise le nombre de routes disjointes par les arêtes entre un ensemble de terminaux. Pour cela nous proposons une heuristique afin de calculer de manière centralisée un ensemble de chemins disjoints par les arêtes entre les terminaux. Les simulations effectuées sur des instances aléatoires montre que cette heuristique donne de bonnes solutions. Enfin, nous adaptons cette heuristique de manière distribuée dans le cas d'un réseau de capteurs ayant de faibles ressources mémoires, une connaissance limitée du réseau et des communications asynchrones. Ce protocole a donné lieu à un brevet international. Pour répondre à la seconde problématique, nous proposons un protocole de routage qui maximise la durée de vie du réseau de capteurs fonctionnant sur batterie. Nous démontrons que ce problème est NP-complet via une réduction polynomiale à partir du problème Partition et proposons un algorithme basé sur un programme linéaire en nombres entiers pour résoudre le problème de routage.

Perspectives

Pour le problème de déploiement de fibres optiques jusqu'au client (FTTH), une des premières choses à faire à la suite de notre travail sera d'expérimenter le modèle et l'algorithme que nous proposons dans cette thèse sur des instances réelles. De cette façon, nous pourrions confronter les solutions obtenues avec notre algorithme avec les méthodes déjà mises en place chez les opérateurs. Une autre voie intéressante serait de faire évoluer notre modèle en intégrant d'autres spécificités du problème rencontré par les opérateurs. Par exemple, il serait intéressant de pouvoir prendre en compte, dans le processus de conception le positionnement des amplificateurs de signal. Ceux-ci sont souvent nécessaires pour assurer une bonne qualité de service dans les zones peu denses. Un autre axe pour la poursuite de ce travail sera d'implémenter une méthode de branch and cut and price. L'implémentation d'un tel algorithme permettrait d'obtenir des solutions optimales pour des instances de petites tailles et de pouvoir mieux évaluer la qualité de l'heuristique que nous proposons.

Pour la prise en compte du délai en fonction de la charge d'utilisation de la liaison dans les réseaux mères des opérateurs de télécommunications, une voie particulièrement intéressante serait de mettre en place une méthode dynamique d'approximation de la fonction convexe de délai. Cet algorithme consisterait à toujours garder une solution réalisable et à affiner dynamiquement l'approximation de la fonction convexe de délai en fonction de la solution courante. Différentes stratégies pourraient être étudiées, par exemple si nous supposons que la famille de chemins supportant le flot est identique entre deux itérations d'approximation. Dans ce cas, considérons une solution optimale dont nous souhaitons affiner l'approximation de la fonction de délai sur l'arc e . Soit ϕ_e le flot supporté par un arc e , D_e son délai et F_e la fonction convexe de délai de l'arc e . Nous pouvons supposer que le flot optimal après l'approximation ϕ_e^* est supérieur ou égal à ϕ_e , car une meilleure approximation du délai relâche les contraintes de délai sur les chemins traversant e et par conséquent autorise plus de flot sur cet arc. Nous pouvons aussi supposer que ϕ_e^* est inférieur ou égal à $F_e^{-1}(D_e)$ le flot maximum sur l'arc e tel que le délai D_e soit atteint par la fonction de délai convexe. Nous pouvons donc espérer améliorer l'approximation en créant un ensemble de fonctions linéaires (défini dans le chapitre 4) approximant la fonction de délai convexe, tel que les points $(\phi_e, F_e(\phi_e))$ et $(F_e^{-1}(D_e), D_e)$ satisfassent à l'égalité deux fonctions linéaires consécutives. Ces points correspondent aux points de cassure de la fonction linéaire par morceaux

(définie par l'ensemble des fonctions linéaires) approximant la fonction convexe.

Pour le premier protocole pour les réseaux de capteurs, il serait intéressant de proposer un algorithme distribué pour les réseaux de capteurs critiques avec des communications synchronisées. En effet, de nombreuses méthodes efficaces de synchronisations ont déjà été étudiées dans [8, 142, 155]. Il semble alors envisageable d'adapter l'algorithme récemment proposé dans [88] qui calcule des plus courts chemins entre chaque paires de terminaux T d'un graphe en $O(|T| + D)$ où D correspond au diamètre du graphe. Une autre voie à explorer est de définir un routage dont le nombre de chemins disjoints par les arêtes est spécifié entre chaque paire de terminaux. Ceci correspond à un problème de multiflot entier. Ce problème est néanmoins difficile à résoudre et à approximer même dans le cas centralisé. Enfin, la prise en compte de la panne d'un capteur en imposant des routages disjoints par les sommets semble très naturelle. Là encore, le problème devient plus difficile à résoudre. En effet, il existe une réduction simple du problème du routage disjoint par les sommets en celui disjoint par les arêtes. Pour cela il suffit d'ajouter un sommet sur chaque arête.

Pour le second protocole de routage, des simulations ainsi qu'une heuristique pour obtenir une solution entière sont des pistes à explorer. De manière équivalente à celle présentée dans le chapitre 3, l'heuristique d'arrondis pour une solution fractionnaire du programme linéaire $R2(t)$ consisterait par exemple pour chaque itération à fixer la valeur d'un chemin à sa partie entière. Nous obtenons alors un nouveau programme linéaire dont nous cherchons la solution optimale par génération de colonnes (branch and price). L'algorithme pourrait privilégier le chemin non fixé dont la valeur fractionnaire est la plus proche de sa partie entière (inférieure). De cette manière, il existe toujours une solution réalisable et le nombre d'itérations est de l'ordre du nombre de demandes totales des capteurs. Enfin, des évolutions du modèle en fonction des besoins des opérateurs peuvent être considérées. Par exemple, la prise en compte de messages descendants émis par les capteurs à intervalles réguliers. Dans ce cas, le routage sera probablement unique entre le capteur et la station de base. Une seconde évolution pourrait être de permettre un routage aller différent du routage retour. La première évolution est possible via une modification mineure du programme linéaire. La seconde évolution impose d'ajouter les chemins retour entre chaque capteur et la station de base à l'ensemble des chemins. Là encore une modification mineure du programme linéaire est à effectuer.

Bibliographie

- [1] Video codec for audiovisual services at p 64 kbits. Technical report, ITU-T, 1993.
- [2] J. Adámek and V. Koubek. Remarks on flows in network with short paths. *Commentationes Mathematicae Universitatis Carolinae*, 12(4) :661–667, 1971.
- [3] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows : Theory, Algorithms, and Applications*. Prentice Hall, 1993.
- [4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications magazine, IEEE*, 40(8) :102–114, 2002.
- [5] I. Alder and S. Cosares. A strongly polynomial algorithm for a special class of linear programs. *Oper. Res.*, 39(6) :955–960, November 1991.
- [6] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. Energy conservation in wireless sensor networks : A survey. *Ad Hoc Networks*, 7(3) :537–568, 2009.
- [7] Y.P. Aneja, V. Aggarwal, and K.P.K. Nair. Shortest chain subject to side constraints. *Networks*, 13(2) :295–302, 1983.
- [8] A. Arenas, A. Díaz-Guilera, J. Kurths, Y. Moreno, and C. Zhou. Synchronization in complex networks. *Physics Reports*, 469(3) :93–153, 2008.
- [9] S. Arora and B. Barak. *Computational complexity : a modern approach*, volume 1. Cambridge University Press Cambridge, 2009.
- [10] G. Baier. *Flows with Path Restriction*. Cuvillier, 2003.
- [11] G. Baier, E. Köhler, and M. Skutella. On the k-splittable flow problem, 2002.
- [12] E. Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic Discrete Methods*, 6(3) :466–486, 1985.

BIBLIOGRAPHIE

- [13] E. Balas. On the convex hull of the union of certain polyhedra. *Operations Research Letters*, 7(6) :279–283, 1988.
- [14] E Balas. Disjunctive programming : Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89(1) :3–44, 1998.
- [15] A. Banerjea. Simulation study of the capacity effects of dispersity routing for fault tolerant realtime channels. *ACM SIGCOMM Computer Communication Review*, 26(4) :194–205, 1996.
- [16] C. Barnhart, C.A. Hane, E.L. Johnson, and G. Sigismondi. A column generation and partitioning approach for multi-commodity flow problems. *Telecommunication Systems*, 3(3) :239–258, 1994.
- [17] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price : Column generation for solving huge integer programs. *Oper. Res*, 1998.
- [18] J.E. Beasley and N. Christofides. An Algorithm for the Resource Constrained Shortest Path Problem. *NETWORKS*, 19 :379–394, 1989.
- [19] S. Beker, N. Puech, and V. Friderikos. A tabu search heuristic for the offline mpls reduced complexity layout design problem. In N.M. Mitrou, K. Kontovasilis, G.N. Rouskas, I. Iliadis, and L. Merakos, editors, *NETWORKING 2004. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications*, volume 3042 of *Lecture Notes in Computer Science*, pages 514–525. Springer Berlin Heidelberg, 2004.
- [20] M. Belaidouni and W. Ben-Ameur. On the minimum cost multiple-source unsplittable flow problem. *RAIRO-Operations Research*, 41(03) :253–273, 2007.
- [21] R. Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8) :716, 1952.
- [22] R. Bellman. On a Routing Problem. *Quarterly of Applied Mathematics*, 16 :87–90, 1958.
- [23] W. Ben-Ameur and A. Ouorou. Mathematical models of the delay constrained routing problem. *Algorithmic Operations Research*, 1(2), 2006.
- [24] C. Bentz, M-C. Costa, L. Létocart, and F. Roupin. A bibliography on multicut and integer multiflow problems. Technical report, Technical Report 654, CEDRIC Centre de Recherche en Informatique du Cnam, <http://cedric.cnam.fr/AfficheMembre.php>, 2004.

-
- [25] C. Berge. Graphes et hypergraphes. 1973.
 - [26] D.P. Bertsekas, R.G. Gallager, and P. Humblet. *Data networks*, volume 2. Prentice-hall Englewood Cliffs, 1987.
 - [27] A. Bley. On the complexity of vertex-disjoint length-restricted path problems. *computational complexity*, 12(3-4) :131–149, 2003.
 - [28] M.L. Bonilla, F.R. Barbosa, and E. Moschim. Techno-economical comparison between gpon and epon networks. In *Innovations for Digital Inclusions, 2009. K-IDI 2009. ITU-T Kaleidoscope* :, pages 1 –5, 31 2009-sept. 1 2009.
 - [29] U. Brandes, G. Neyer, and D. Wagner. Edge-disjoint paths in planar graphs with short total length. *Konstanzer Schriften in Mathematik und Informatik*, 19, 1996.
 - [30] S. Chamberland. Designing low cost access networks with iptv performance constraints. In *Next Generation Internet Networks, 2008. NGI 2008*, pages 45 –52, april 2008.
 - [31] M. Chardy, M-C. Costa, A. Faye, and M. Trampont. Optimizing splitter and fiber location in a multilevel optical ftth network. *European Journal of Operational Research*, 222(3) :430–440, 2012.
 - [32] M. Chardy and C. Hervet. FTTH network design under O&AM constraints. In *Proceedings of the 5th international conference on Network optimization, INOC'11*, pages 98–104, Berlin, Heidelberg, 2011. Springer-Verlag.
 - [33] X. Cheng, Ding-Zhu Du, L. Wang, and B. Xu. Relay sensor placement in wireless sensor networks. *Wireless Networks*, 14(3) :347–355, 2008.
 - [34] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*, pages 51–229, 2006.
 - [35] E. Cohen and N. Megiddo. Improved algorithms for linear inequalities with two variables per inequality. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing, STOC '91*, pages 145–155, New York, NY, USA, 1991. ACM.
 - [36] E. Cohen and N. Megiddo. New algorithms for generalized network flows. *Mathematical Programming*, 64 :325–336, 1994. 10.1007/BF01582579.
 - [37] C.R. Collins and K. Stephenson. A circle packing algorithm. *Computational Geometry*, 25(3) :233 – 256, 2003.
 - [38] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.
-

BIBLIOGRAPHIE

- [39] J.R. Correa, A.S. Schulz, and N.E. Stier-Moses. Fast, fair, and efficient flows in networks. *Operations Research*, 55(2) :215–225, 2007.
- [40] M-C. Costa, L. Létocart, and R. Roubin. Minimal multicut and maximal integer multiflow : a survey. *European Journal of Operational Research*, 162(1) :55–69, 2005.
- [41] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A framework for qos-based routing in the internet. Technical report, RFC, 1998.
- [42] G.B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. *New York*, 1951.
- [43] G.B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1) :101–111, 1960.
- [44] B. Deb, S. Bhatnagar, and B. Nath. Reinform : Reliable information forwarding using multiple paths in sensor networks. In *Local Computer Networks, 2003. LCN'03. Proceedings. 28th Annual IEEE International Conference on*, pages 406–415. IEEE, 2003.
- [45] J.M. Van den Akker, C.A.J. Hurkens, and M.W.P. Savelsbergh. Time-indexed formulations for machine scheduling problems : Column generation. *INFORMS Journal on Computing*, 12(2) :111–124, 2000.
- [46] G. Desaulniers, J. Desrosiers, I. Ioachim, M.M. Solomon, F. Soumis, and D. Villeneuve. A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems. *Fleet Management and Logistics*, pages 57–93, 1998.
- [47] G. Desaulniers, J. Desrosiers, and M.M. Solomon. *Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems*. Springer, 2002.
- [48] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2) :pp. 342–354, 1992.
- [49] M. Desrochers and F. Soumis. A generalized permanent labeling algorithm for the shortest path problem with time windows. *Information Systems and Operations Research*, 26 :191–212, 1988.
- [50] J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis. Chapter 2 time constrained routing and scheduling. In C.L. Monma M.O. Ball, T.L. Magnanti and G.L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 35 – 139. Elsevier, 1995.

- [51] J. Desrosiers, F. Soumis, and M. Desrochers. Plus court chemin avec contraintes d'horaires. *RAIRO*, 17 :357–377, 1983.
- [52] J. Desrosiers, F. Soumis, and M. Desrochers. Routing with time windows by column generation. *Networks*, 14(4) :545–565, 1984.
- [53] J. Desrosiers, F. Soumis, M. Desrochers, and M. Sauvé Gerad. Methods for routing with time windows. *European Journal of Operational Research*, 23(2) :236 – 245, 1986.
- [54] R. Diestel. Graph theory. 2005, 2005.
- [55] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–271, 1959.
- [56] J. Elam, F. Glover, and D. Klingman. A strongly convergent primal simplex algorithm for generalized networks. *Mathematics of Operations Research*, 4(1) :39–59, 1979.
- [57] P. Elias, A. Feinstein, and C. Shannon. A note on the maximum flow through a network. *Information Theory, IRE Transactions on*, 2(4) :117–119, 1956.
- [58] S.C. Ergen and P. Varaiya. Optimal placement of relay nodes for energy efficiency in sensor networks. In *Communications, 2006. ICC'06. IEEE International Conference on*, volume 8, pages 3473–3479. IEEE, 2006.
- [59] S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. In *Foundations of Computer Science, 1975., 16th Annual Symposium on*, pages 184–193. IEEE, 1975.
- [60] G. Exoo. On line disjoint paths of bounded length. *Discrete Mathematics*, 44(3) :317–318, 1983.
- [61] L.K. Fleischer and K.D. Wayne. Fast and simple approximation schemes for generalized flow. *Proceedings of the 10th annual ACM-SIAM Symposium on Discrete Mathematics*, 1999.
- [62] R.W. Floyd. Algorithm 97 : Shortest path. *Commun. ACM*, 5(6) :345–, June 1962.
- [63] L.R. Ford and D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3) :399–404, 1956.
- [64] L.R. Ford and D.R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5(1) :97–101, 1958.

BIBLIOGRAPHIE

- [65] A. Frangioni and G. Gallo. A bundle type dual-ascent approach to linear multicommodity min-cost flow problems. *INFORMS Journal on Computing*, 11(4) :370–393, 1999.
- [66] L. Fratta, M. Gerla, and L. Kleinrock. The flow deviation method : An approach to store-and-forward communication network design. *Networks*, 3(2) :97–133, 1973.
- [67] J. Galtier and A. Laugier. Flow on data network and a positive semidefinite representable delay function. *Journal of Interconnection Networks*, 8(01) :29–43, 2007.
- [68] Q. Gao, K.J. Blow, D.J. Holding, and I. Marshall. Analysis of energy conservation in sensor networks. *Wireless Networks*, 11(6) :787–794, 2005.
- [69] M.R. Garey and D.S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.
- [70] M. Gerla and L. Kleinrock. On the topological design of distributed computer networks. *Communications, IEEE Transactions on*, 25(1) :48 – 60, jan 1977.
- [71] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6) :849–859, 1961.
- [72] F. Glover, J. Hultz, D. Klingman, and J. Stutz. Generalized networks : A fundamental computer-based planning tool. *Management Science*, 24(12) :1209–1220, 1978.
- [73] F. Glover and D. Klingman. The generalized lattice-point problem. *Operations Research*, 21(1) :141–155, 1973.
- [74] A.V. Goldberg, S.A Plotkin, and volume=16 number=2 pages=351–381 year=1991 publisher=INFORMS E Tardos, journal=Mathematics of Operations Research. Combinatorial algorithms for the generalized circulation problem.
- [75] A.V. Goldberg and R.E Tarjan. Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM (JACM)*, 36(4) :873–886, 1989.
- [76] I.E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3(3) :227–252, 2002.
- [77] M. Grötschel, L. Lovász, and A Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, Berlin.
- [78] M. Grötschel, L. Lovász, and A Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2) :169–197, 1981.

-
- [79] M. Grötschel, L. Lovász, and A. Schrijver. Polynomial algorithms for perfect graphs. *Ann. Discrete Math*, 21 :325–356, 1984.
- [80] S. Gualandi, F. Malucelli, and D. Sozzi. On the design of the next generation access networks. In Andrea Lodi, Michela Milano, and Paolo Toth, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 6140 of *Lecture Notes in Computer Science*, pages 162–175. Springer Berlin / Heidelberg.
- [81] V Guruswami, S Khanna, R Rajaraman, B Shepherd, and M Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *Journal of Computer and System Sciences*, 67(3) :473–496, 2003.
- [82] J. Halpern and I. Priess. Shortest path with time constraints on movement and parking. *Networks*, 4(3) :241–253, 1974.
- [83] G.Y. Handler and I. Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10(4) :293–309, 1980.
- [84] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1) :36–42, 1992.
- [85] H. Hijazi, P. Bonami, G. Cornuéjols, and A. Ouorou. Mixed-integer nonlinear programs featuring "on/off" constraints. *Comput. Optim. Appl.*, 52(2) :537–558, June 2012.
- [86] D.S. Hochbaum. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM J. Comput.*, 23(6) :1179–1192, December 1994.
- [87] K. Holmberg and D. Yuan. A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS Journal on Computing*, 15(1) :42–57, 2003.
- [88] S. Holzer and R. Wattenhofer. Optimal distributed all pairs shortest paths and applications. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, pages 355–364. ACM, 2012.
- [89] D.F. Hsu. On container width and length in graphs, groups, and networks. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 77(4) :668–680, 1994.
- [90] S. Imbrosciano and A. Laugier. Calcul de routages assurant la durée de vie maximum d'un réseau de capteurs. *Dépôt de brevet*, 2010.

BIBLIOGRAPHIE

- [91] S. Imbrosciano and A. Laugier. Communication de données dans un réseau de capteurs. *World International Property Organization*, WO/2012/131280.
- [92] S. Imbrosciano, A. Laugier, and Y. Vaxès. Un algorithme pour le déploiement des réseaux FTTH. 2011.
- [93] I. Ioachim, S. Gélinas, F. Soumis, and J. Desrosiers. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31(3) :193–204, 1998.
- [94] A. Itai, Y. Perl, and Y. Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks*, 12(3) :277–286, 1982.
- [95] B. Jaumard and R. Chowdhury. Location and allocation of switching equipment (splitters/awgs) in a wdm pon network. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–8, 31 2011-aug. 4 2011.
- [96] B. Jaumard, F. Semet, and T. Vovor. A two-phase resource constrained shortest path algorithm for acyclic graphs. *Cahiers du GERAD*, 1996.
- [97] B. Jaumard, F. Semet, and T. Vovor. A generalized linear programming model for nurse scheduling. *European journal of operational research*, 107(1) :1–18, 1998.
- [98] B. Kantarci and M.T. Hussein. Availability and cost constrained fast planning of passive optical networks under various survivability policies. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 260–263, oct. 2010.
- [99] M. Kantor, K. Wajda, B. Lannoo, K. Casier, S. Verbrugge, M. Pickavet, L. Wosinska, Jiajia Chen, and A. Mitcsenkov. General framework for techno-economic analysis of next generation access networks. In *Transparent Optical Networks (ICTON), 2010 12th International Conference on*, pages 1–4, 27 2010-july 1 2010.
- [100] R.M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.
- [101] C. Kaushal. Resource allocation and capacity assignment in distributed systems. *Computers and Operations Research*, 23(11) :1025–1041, 1996.
- [102] J.C.M. Keijsper, R.A. Pendavingh, and L. Stougie. A linear programming formulation of mader’s edge-disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 96(1) :159–163, 2006.

-
- [103] J. Jr. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4) :703–712, 1960.
 - [104] L.G Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady* 20, pages 191–194, 1971.
 - [105] M. Klein. A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, 14(3) :205–220, 1967.
 - [106] L. Kleinrock. *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, 1975.
 - [107] J.G. Klincewicz. Optimization issues in quality of service. In R.G.C. Mauricio and P.M. Pardalos, editors, *Handbook of Optimization in Telecommunications*, pages 435–458. Springer US, 2006.
 - [108] D.E. Knuth. Big omicron and big omega and big theta. *ACM Sigact News*, 8(2) :18–24, 1976.
 - [109] P. Kolman and C. Scheideler. Improved bounds for the unsplittable flow problem. *Journal of Algorithms*, 61(1) :20–44, 2006.
 - [110] P. Kourtessis, C. Almeida, C.H. Chang, J. Chen, S. Di Bartolo, P. Fasser, F. Gagnaire, E. Leitgeb, M. Lima, M. Loschnigg, M. Marciniak, N. Pavlovic, Y. Shachaf, A. Teixeira, G. Tosi Beleffi, and L. Wosinska. Evolution of optical access networks. In Ioannis Tomkos, Maria Spyropoulou, Karin Ennser, Martin Kohn, and Branko Mikac, editors, *Towards Digital Optical Networks*, volume 5412 of *Lecture Notes in Computer Science*, pages 97–131. Springer Berlin / Heidelberg, 2009.
 - [111] C. Kwang-Ting and F.Y.-S. Lin. Minimax end-to-end delay routing and capacity assignment for virtual circuit networks. In *Global Telecommunications Conference, 1995. GLOBECOM '95., IEEE*, volume 3, pages 2134–2138 vol.3, nov 1995.
 - [112] B. Lakic and M. Hajduczenia. On optimized passive optical network (pon) deployment. *Proceedings of the AccessNets 2007 Second International Conference on Access Networks*, 2007.
 - [113] S. Lallukka and P. Raatikainen. Link utilization and comparison of epon and gpon access network cost. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 1, page 5 pp., nov.-2 dec. 2005.
 - [114] E.L. Lawler. Combinatorial optimization : Networks and matroids. 1976. *Holt, Rinehart and Winston, New York*.

BIBLIOGRAPHIE

- [115] L.J. Leblanc. *Mathematical programming algorithms for large scale network equilibrium and network design problems*. PhD thesis, Evanston, IL, USA, 1973. AAI7407771.
- [116] Ji Li and Gangxiang Shen. Cost minimization planning for passive optical networks. In *National Fiber Optic Engineers Conference*, page NThD1. Optical Society of America, 2008.
- [117] M. De Andrade C. Cervello-Pastor S. Sallent Ll. Gutierrez, P. Garfias. Trends in telecommunications technologies. 2010-03-01.
- [118] L. Lovász, V. Neumann-Lara, and M. Plummer. Mengerian theorems for paths of bounded length. *Periodica Mathematica Hungarica*, 9(4) :269–276, 1978.
- [119] Miao Lv and Xue Chen. Heuristic based multi-hierarchy passive optical network planning. In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, pages 1–4, sept. 2009.
- [120] A.R. Mahjoub and S.T. McCormick. Max flow and min cut with bounded-length paths : complexity, algorithms, and approximation. *Mathematical programming*, 124(1-2) :271–284, 2010.
- [121] K. Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10(1) :96–115, 1927.
- [122] A. Mingozzi, M.A. Boschetti, S. Ricciardelli, and L. Bianco. A set partitioning approach to the crew scheduling problem. *Operations Research*, 47(6) :873–888, 1999.
- [123] A. Mitcsenkov, G. Paksy, and T. Cinkler. Geography- and infrastructure-aware topology design methodology for broadband access networks (fttx). *Photonic Network Communications*, 21 :253–266, 2011. 10.1007/s11107-010-0297-4.
- [124] K. Onaga. Dynamic programming of optimum flows in lossy communication nets. *Circuit Theory, IEEE Transactions on*, 13(3) :282–287, 1966.
- [125] K. Onaga. Optimum flows in general communication networks. *Journal of the Franklin Institute*, 283(4) :308–327, 1967.
- [126] T. Orphanoudakis, H.-C. Leligou, E. Kosmatos, and J. D. Angelopoulos. Performance evaluation of gpon vs epon for multi-service access. *Int. J. Commun. Syst.*, 22(2) :187–202, February 2009.

-
- [127] F. Othman, N. Bouabdallah, and R. Boutaba. Energy conservation in reliable wireless sensor networks.
 - [128] A. Ouorou. *Décomposition proximale des problèmes de multiflot à critère convexe. Application aux problèmes de routage dans les réseaux de communication*. PhD thesis, 1995.
 - [129] A. Ouorou, P. Mahey, and J.-Ph. Vial. A survey of algorithms for convex multicommodity flow problems. *Management Science*, 1997.
 - [130] G. Owen. Cutting planes for programs with disjunctive constraints. *Journal of Optimization Theory and Applications*, 11(1) :49–55, 1973.
 - [131] C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization : algorithms and complexity*. Courier Dover Publications, 1998.
 - [132] K.F. Poon, D.B. Mortimore, and J. Mellis. Designing optimal ftth and pon networks using new automatic methods. In *Access Technologies, 2006. The 2nd Institution of Engineering and Technology International Conference on*, pages 49 – 52, june 2006.
 - [133] W.B. Powell and Z.L. Chen. A generalized threshold algorithm for the shortest path problem with time windows. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 40 :303–318, 1998.
 - [134] M. Queiroz and C. Jr. Humes. A heuristic for the continuous capacity and flow assignment. *European Journal of Operational Research*, 146(3) :444 – 459, 2003.
 - [135] C.C. Ribeiro and M. Minoux. A heuristic approach to hard constrained shortest path problems. *Discrete Applied Mathematics*, 10(2) :125 – 137, 1985.
 - [136] D. Ronen and Y. Perl. Heuristics for finding a maximum number of disjoint bounded paths. *Networks*, 14(4) :531–544, 1984.
 - [137] A. Schrijver. A short proof of Mader’s y-paths theorem. *Journal of Combinatorial Theory Series B*, 82(2) :319–321, 2001.
 - [138] A. Sebő and L. Szegő. The path-packing structure of graphs. In *Integer Programming and Combinatorial Optimization*, pages 256–270. Springer, 2004.
 - [139] H.D. Sherali and W.P. Adams. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics*, 52(1) :83–106, 1994.
 - [140] C. Shigang and K. Nahrstedt. An overview of quality of service routing for next-generation high-speed networks : problems and solutions. *Network, IEEE*, 12(6) :64 –79, nov/dec 1998.
-

BIBLIOGRAPHIE

- [141] M. Sipser. *Introduction to the Theory of Computation*, volume 27. Thomson Course Technology Boston, MA, 2006.
- [142] F. Sivrikaya and B. Yener. Time synchronization in sensor networks : a survey.
- [143] A. Srinivas and E. Modiano. Minimum energy disjoint path routing in wireless ad-hoc networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 122–133. ACM, 2003.
- [144] K. Stephenson, J.W. Cannon, W.J. Floyd, and W.R. Parry. Introduction to circle packing : the theory of discrete analytic functions. *The Mathematical Intelligencer*, 29 :63–66, 2007.
- [145] K. Truemper. On max flows with gains and pure min-cost flows. *SIAM Journal on Applied Mathematics*, 32(2) :450–456, 1977.
- [146] J. Truffot, C. Duhamel, and P. Mahey. k-splittable delay constrained routing problem : A branch and price approach. In *Design and Reliable Communication Networks, 2007. DRCN 2007. 6th International Workshop on*, pages 1–8, oct. 2007.
- [147] H. van der Holst and J.C. De Pina. Length-bounded disjoint paths in planar graphs. *Discrete Applied Mathematics*, 120(1) :251–261, 2002.
- [148] C.A. van Eijl. Capacity planning for carrier-scale ip networks. *BT Technology Journal*, 20(3) :116–123, July 2002.
- [149] D. Wagner and K. Weihe. A linear-time algorithm for edge-disjoint paths in planar graphs. *Combinatorica*, 15(1) :135–150, 1995.
- [150] A. Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1) :70–79, January/February 1987.
- [151] D.P. Williamson and D.B. Shmoys. *The design of approximation algorithms*, chapter 1. Cambridge University Press, 2011.
- [152] Hong-Hsu Yen and F.Y.-S. Lin. Backbone network design with qos requirements. In *ICN (2)*, pages 148–157, 2001.
- [153] Hong-Hsu Yen and F.Y.-S. Lin. Near-optimal delay constrained routing in virtual circuit networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 750–756 vol.2, 2001.
- [154] J.Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11) :712–716, 1971.

- [155] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer networks*, 52(12) :2292–2330, 2008.

Annexes

Annexe 1 : Brevet

(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la
Propriété Intellectuelle
Bureau international



WIPO | PCT



(10) Numéro de publication internationale

WO 2012/131280 A1

(51) Classification internationale des brevets :
H04L 12/56 (2006.01) H04W 84/18 (2009.01)

(21) Numéro de la demande internationale :
PCT/FR2012/050712

(22) Date de dépôt international :
2 avril 2012 (02.04.2012)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
1152737 31 mars 2011 (31.03.2011) FR

(71) Déposant (pour tous les États désignés sauf US) :
FRANCE TELECOM [FR/FR]; 78 Rue Olivier de Serres,
F-75015 Paris (FR).

(72) Inventeurs; et

(75) Inventeurs/Déposants (pour US seulement) : LAUGIER,
Alexandre [FR/FR]; 2 place André Laugier, F-06440
Peille (FR). IMBROSCIANO, Sébastien [FR/FR]; 21
avenue Fabron -, Résidence Fabron Park, F-06200 Nice
(FR).

(74) Mandataire : FRANCE TELECOM R&D/PIV/BRE-
VETS; CASPAR Catherine, 38-40 rue du Général Leclerc,
F-92794 Issy Moulineaux Cedex 9 (FR).

(81) États désignés (sauf indication contraire, pour tout titre
de protection nationale disponible) : AE, AG, AL, AM,
AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ,
CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO,
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,
HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR,
KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME,
MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ,
OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD,
SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR,
TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) États désignés (sauf indication contraire, pour tout titre
de protection régionale disponible) : ARIPO (BW, GH,
GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ,
UG, ZM, ZW), eurasién (AM, AZ, BY, KG, KZ, MD, RU,
TJ, TM), européen (AL, AT, BE, BG, CH, CY, CZ, DE,
DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU,
LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK,
SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ,
GW, ML, MR, NE, SN, TD, TG).

Déclarations en vertu de la règle 4.17 :

— relative à la qualité d'inventeur (règle 4.17.iv))

Publiée :

— avec rapport de recherche internationale (Art. 21(3))

(54) Title : DATA COMMUNICATION IN A SENSOR ARRAY

(54) Titre : COMMUNICATION DE DONNÉES DANS UN RÉSEAU DE CAPTEURS

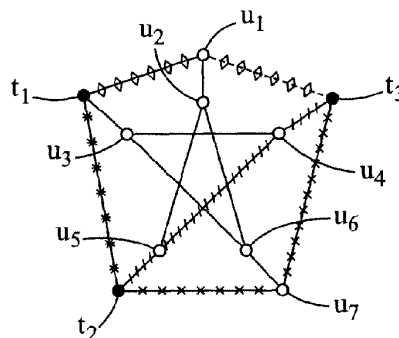


FIG. 10

(57) Abstract : The invention relates to processing data for data communication in an array, the array including a set of devices and a set of links, each link connecting two of said devices. The data-processing includes: iteratively determining a set of communication paths between the devices, each communication path comprising one or more link(s) from the set of links, and configuring the devices such that each communication between two devices uses a communication path from the set of communication paths.

(57) Abrégé : L'invention se rapporte à un traitement de données pour la communication de données dans un réseau, le réseau comprenant un ensemble de dispositifs et un ensemble de liaisons, chaque liaison reliant deux desdits dispositifs. Le traitement de données comprenant : - la détermination, de manière itérative, d'un ensemble de chemins de communication entre les dispositifs, chaque chemin de communication comprenant une ou plusieurs liaison(s) de l'ensemble de liaisons, et - la configuration des dispositifs pour que chaque communication entre deux dispositifs utilise un chemin de communication de l'ensemble de chemins de communication.

COMMUNICATION DE DONNEES DANS UN RESEAU DE CAPTEURS

La présente invention vise un traitement de données pour la communication de données dans un réseau informatique. En particulier, la présente invention vise un traitement de données
5 pour la communication de données dans un réseau de capteurs.

Les capteurs actuels, par leur diversité, sont capables de mesurer différentes grandeurs physiques ou chimiques, par exemple une position d'un objet à la surface de la terre, une température, un niveau d'eau, un niveau de lumière, une présence de certains corps chimiques, etc. Certains capteurs sont en outre capables de fournir des images.

10 De manière générale, on appelle capteur un dispositif capable de réaliser une mesure d'un paramètre de son environnement, et de transmettre des données de mesure correspondantes.

On appelle réseau de capteurs un réseau informatique comportant un ensemble de capteurs et un ensemble de voies de communication permettant aux capteurs de transmettre des données de mesure. Un réseau de capteurs peut en outre comporter d'autres dispositifs, par exemple un ou
15 plusieurs dispositif(s) de collecte, configuré(s) pour collecter des données de mesure fournies par les capteurs, et /ou un ou plusieurs dispositif(s) de routage, configuré(s) pour permettre le routage des données de mesure transmises. Les voies de communication sont par exemple des liaisons radioélectriques ou des liaisons filaires.

Actuellement, les capteurs sont généralement uniquement configurés pour mesurer un
20 paramètre de l'environnement dans lequel ils sont implantés et pour transmettre les données de mesure correspondantes vers un point de collecte où l'information sera traitée. Les communications vers ce point de collecte se font par exemple suivant des chemins de communication définissant une topologie virtuelle arborescente. La sécurité des communications peut alors être assurée par le fait que l'arbre virtuel n'est pas unique.

25 Cependant, cette solution n'est pas applicable dans le cas d'un réseau de capteurs comportant des capteurs en outre configurés pour agir sur leur environnement, seul ou en association avec d'autres dispositifs. En effet, dans ce cas, le problème n'est plus d'assurer la sécurité des communications vers un point collecte, mais d'assurer la sécurité des communications entre les différents capteurs du réseau de capteurs.

30 La présente invention vient améliorer la situation.

A cet effet, l'invention propose un procédé de traitement de données pour la communication de données dans un réseau. Le réseau comprend un ensemble de dispositifs formant

nœuds dudit réseau et un ensemble de liaisons, chaque liaison reliant deux desdits dispositifs. Le procédé comprend des étapes :

- déterminer un ensemble de chemins de communication entre les dispositifs, chaque chemin de communication comprenant une ou plusieurs liaison(s) de l'ensemble de liaisons,
- 5 - configurer les dispositifs pour que chaque communication entre deux dispositifs utilise un chemin de communication de l'ensemble de chemins de communication.

Les communications entre les dispositifs nécessitent la détermination préalable des chemins suivant lesquels ces communications pourront être acheminées : cela revient à contraindre les dispositifs à utiliser un chemin de cet ensemble. Cet ensemble de chemins représente donc les
10 chemins « autorisés » à un instant donné.

Cette détermination préalable permet en outre de déterminer un nombre maximum de chemins susceptibles d'être utilisés à un instant donné, ce qui permet de maximiser la bande passante totale utilisable dans le réseau.

Avantageusement, les chemins de communication de l'ensemble de chemins de
15 communication sont disjoints, c'est-à-dire que deux chemins quelconques de cet ensemble ne possèdent aucune liaison en commun (mais peuvent comporter un ou plusieurs nœuds en commun). L'utilisation de chemins disjoints permet d'augmenter la sécurité des communications entre les dispositifs du réseau. En effet, l'utilisation d'un ensemble de chemins disjoints améliore la robustesse aux pannes, puisque, en cas de panne affectant une liaison, un seul chemin sera affecté.
20 En outre, cela permet d'éviter les risques de collision, donc les risques de non-transmission de données ou d'erreurs de transmission.

L'étape de détermination de l'ensemble de chemins de communication peut être mise en œuvre dans un des dispositifs. L'étape de configuration peut alors comporter une opération de transmission par le dispositif, à destination d'au moins un autre dispositif du réseau, d'un message
25 contenant des informations relatives à l'ensemble de chemins de communication.

L'étape de détermination de l'ensemble de chemins de communication peut comporter une opération de sélection de deux dispositifs parmi les dispositifs du réseau, et une opération de détermination d'un nouveau chemin de communication entre les deux dispositifs sélectionnés.

L'opération de détermination d'un nouveau chemin de communication entre les deux
30 dispositifs sélectionnés peut comprendre la sélection d'une liaison de l'ensemble de liaisons, et la construction d'un nouveau chemin de communication en utilisant la liaison sélectionnée, un plus court chemin entre un des deux dispositifs sélectionnés et une première extrémité de la liaison, et un plus court chemin entre la deuxième extrémité de la liaison et l'autre des deux dispositifs sélectionnés.

Les liaisons peuvent être des liaisons radioélectriques. En variante, les liaisons peuvent être des liaisons filaires.

L'invention propose également un programme informatique comportant des instructions pour la mise en œuvre du procédé décrit ci-dessus lorsque ce programme est exécuté par un processeur.

L'invention propose également un dispositif, conçu pour être utilisé dans un réseau comprenant un ensemble de dispositifs et un ensemble de liaisons, chaque liaison reliant deux desdits dispositifs, ledit dispositif étant configuré pour :

- déterminer un ensemble de chemins de communication à utiliser entre les dispositifs du réseau, chaque chemin de communication comprenant une ou plusieurs liaison(s) de l'ensemble de liaisons,
- configurer les dispositifs du réseau pour que chaque communication entre deux desdits dispositifs utilise un chemin de communication dudit ensemble de chemins de communication.

Le dispositif peut être un capteur configuré pour transmettre des données de mesure, correspondant à une mesure réalisée par le capteur, à au moins un autre capteur du réseau, le capteur étant en outre configuré pour que chaque communication entre ledit capteur et un autre capteur du réseau utilise au moins un chemin de communication de l'ensemble de chemins de communication.

L'invention propose également un réseau comprenant un ensemble de dispositifs et un ensemble de liaisons, chaque liaison reliant deux desdits dispositifs. Au moins un des dispositifs est configuré pour déterminer un ensemble de chemins de communication entre les dispositifs, chaque chemin de communication comprenant une ou plusieurs liaison(s) de l'ensemble de liaisons. Les dispositifs sont en outre configurés pour que chaque communication entre deux des dispositifs utilise un chemin de communication de l'ensemble de chemins de communication.

L'ensemble de dispositifs peut comprendre des capteurs et des dispositifs de routage.

Le dispositif réalisant la détermination peut être un capteur en outre configuré pour transmettre, à destination d'au moins un dispositif de routage, un message contenant des informations relatives à l'ensemble de chemins de communication.

Le capteur réalisant la détermination peut en outre être configuré pour sélectionner deux capteurs parmi les capteurs du réseau de capteurs, et pour déterminer un nouveau chemin de communication entre les deux capteurs sélectionnés.

Le capteur réalisant la détermination peut en outre être configuré pour sélectionner une liaison de l'ensemble de liaisons, et pour construire un nouveau chemin de communication à partir de la liaison sélectionnée, d'un plus court chemin entre un des deux capteurs sélectionnés et une

première extrémité de la liaison, et d'un plus court chemin entre la deuxième extrémité de la liaison et l'autre des deux capteurs sélectionnés.

D'autres caractéristiques et avantages de l'invention apparaîtront encore à la lecture de la description qui va suivre. Celle-ci est purement illustrative et doit être lue en regard des dessins annexés sur lesquels :

- la Figure 1 est un graphe représentant un réseau de capteurs ;
- la Figure 2 est un organigramme illustrant les étapes d'un procédé de communication de données dans le réseau de capteurs selon un mode de réalisation de l'invention, cet organigramme pouvant représenter l'algorithme général du programme informatique au sens de l'invention ;
- les Figures 3 à 10 sont des vues similaires à la figure 1 illustrant le déroulement de différentes étapes du procédé de la figure 2 ; et
- la Figure 11 est un schéma fonctionnel montrant un capteur du réseau de capteurs, selon un mode de réalisation de l'invention.

L'invention concerne un procédé de traitement de données pour la communication de données dans un réseau d'entités communicantes. Elle est décrite dans le cas de son application à un réseau de capteurs.

La figure 1 montre un réseau de capteurs G représenté sous la forme d'un graphe G.

Le réseau de capteurs G est par exemple utilisé dans le domaine de la sécurité, notamment pour détecter une situation anormale (intrusion d'une personne, dégradation d'une installation, alerte météorologique, détection précoce de séismes).

En variante, le réseau de capteurs G peut être utilisé dans le domaine de la surveillance de l'environnement, notamment pour fournir des informations sur l'environnement (analyse de l'air en des points particuliers, analyse de l'eau, détection des incendies de forêt) ou pour déterminer des assolements optimaux pour différentes cultures.

En variante, le réseau de capteurs G peut être utilisé dans le domaine des applications relatives à la chaîne support logistique, notamment pour délivrer la position d'un véhicule, ou de marchandises, afin de suivre les acheminements et/ou de déterminer le niveau des stocks.

En variante, le réseau de capteurs G peut être utilisés dans le domaine militaire, notamment pour surveiller un champ d'opérations et des actions logistiques devant subvenir aux besoins des opérations, ainsi que dans le domaine de la productique et la domotique.

Le réseau de capteurs G comprend un ensemble de dispositifs représentés respectivement par des nœuds (ou sommets) du graphe. On appelle ensemble V l'ensemble des nœuds du graphe G.

L'ensemble de dispositifs comporte trois capteurs t_1 , t_2 et t_3 . Les capteurs t_1 , t_2 et t_3 sont configurés pour mesurer un paramètre de l'environnement dans lequel ils sont implantés et pour
5 transmettre des données de mesure correspondantes à destination d'un autre dispositif du réseau de capteurs G. Chaque capteur t_1 , t_2 , t_3 est en outre configuré pour agir sur son environnement, seul ou en association avec un autre dispositif, en fonction de données de mesure mesurées par le capteur t_1 , t_2 , t_3 ou reçues d'un autre capteur t_1 , t_2 , t_3 .

En se référant à la figure 11, le capteur t_1 comporte un module de mesure 1 configuré pour
10 mesurer un paramètre de son environnement, et un module d'émission-réception 2 configuré pour transmettre des données de mesure correspondant à une mesure réalisée par le module de mesure 1 et pour recevoir des données, notamment des données de mesure, provenant des capteurs t_2 et t_3 . Le capteur t_1 comporte en outre un module de traitement 3 configuré pour mettre en œuvre un procédé de communication de données dans le réseau de capteurs G, décrit en détails plus loin. Les capteurs
15 t_2 et t_3 sont réalisés de manière similaire.

L'ensemble de dispositifs comporte également sept dispositifs de routage u_1 à u_7 . Les dispositifs de routage u_1 à u_7 sont configurés pour router les communications entre les capteurs t_1 , t_2 et t_3 .

Ainsi, les capteurs t_1 à t_3 sont des dispositifs émetteurs et/ou destinataires d'un paquet de
20 données, tandis que les dispositifs de routage u_1 à u_7 sont des dispositifs servant d'intermédiaires pour l'acheminement d'un tel paquet de données d'un capteur à un autre. Les capteurs du réseau G sont reliés entre eux par un ensemble de voies de communication (appelées ici également chemins de communication), comprenant des liaisons, chaque liaison (a,b) entre deux dispositifs a et b étant représentée par un lien (ou arêtes) du graphe G. On appelle ensemble E l'ensemble des liens /
25 arêtes du graphe G.

Sur l'exemple représenté à la figure 1, l'ensemble de voies de communication comporte :

- une liaison (t_1, u_1) entre le capteur t_1 et le dispositif de routage u_1 ,
- une liaison (t_1, u_3) entre le capteur t_1 et le dispositif de routage u_3 ,
- une liaison (t_1, t_2) entre le capteur t_1 et le capteur t_2 ,
- 30 - une liaison (t_2, u_5) entre le capteur t_2 et le dispositif de routage u_5 ,
- une liaison (t_2, u_7) entre le capteur t_2 et le dispositif de routage u_7 ,
- une liaison (t_3, u_1) entre le capteur t_3 et le dispositif de routage u_1 ,
- une liaison (t_3, u_4) entre le capteur t_3 et le dispositif de routage u_4 ,

- une liaison (t_3, u_7) entre le capteur t_3 et le dispositif de routage u_7 ,
- une liaison (u_1, u_2) entre le dispositif de routage u_1 et le dispositif de routage u_2 ,
- une liaison (u_2, u_5) entre le dispositif de routage u_2 et le dispositif de routage u_5 ,
- une liaison (u_2, u_6) entre le dispositif de routage u_2 et le dispositif de routage u_6 ,
- 5 - une liaison (u_3, u_4) entre le dispositif de routage u_3 et le dispositif de routage u_4 ,
- une liaison (u_3, u_6) entre le dispositif de routage u_3 et le dispositif de routage u_6 ,
- une liaison (u_4, u_5) entre le dispositif de routage u_4 et le dispositif de routage u_5 , et
- une liaison (u_6, u_7) entre le dispositif de routage u_6 et le dispositif de routage u_7 .

Les liaisons sont des liaisons radioélectriques, par exemple des liaisons de type Wifi. En
 10 variante, les liaisons pourraient être des liaisons filaires. Une liaison (a, b) est considérée sans orientation, c'est-à-dire qu'elle sert à la communication d'un dispositif a vers un dispositif b, et à la communication du dispositif b vers le dispositif a.

On appelle sous-ensemble T de l'ensemble V ($T \subseteq V$) le sous-ensemble des nœuds du
 graphe G représentant des dispositifs configurés pour pouvoir communiquer entre eux, qui sont
 15 émetteurs et/ou destinataires d'un paquet de données, et non pas simples intermédiaires dans l'acheminement de paquets. En d'autres termes, le sous-ensemble T comporte uniquement les capteurs t_1, t_2, t_3 , mais pas les dispositifs de routage.

On appelle sous-ensemble $(V-T)$ de l'ensemble V le sous-ensemble des nœuds du graphe G
 qui n'appartiennent pas au sous-ensemble T . Le sous-ensemble $(V-T)$ comporte donc les dispositifs
 20 de routage u_1 à u_7 .

Une communication entre deux capteurs t peut passer par un ou plusieurs dispositifs de
 routage u , mais ne peut pas passer par un troisième capteur t . Ainsi, on fait l'hypothèse ici qu'un
 chemin de communication joignant un premier capteur t et un deuxième capteur t' du sous-
 ensemble T ne peut pas contenir un autre capteur du sous-ensemble T . Ainsi, les chemins de
 25 communication entre deux capteurs t et t' du sous-ensemble T ne comprendront aucun capteur en commun, si ce n'est les capteurs t et t' .

On considère par exemple que les liaisons radioélectriques présentent une capacité unitaire
 donnée et tous les dispositifs émettent sur une même fréquence. En conséquence, une seule
 communication peut être acheminée à un instant donné sur une liaison.

30 Les dispositifs n'ont pas nécessairement connaissance de l'ensemble de la topologie du réseau de capteurs. Selon des modes de réalisation de l'invention, les dispositifs t, u ont

connaissance de leurs voisins, c'est-à-dire des dispositifs qui leurs sont directement reliés par une liaison.

Selon des modes de réalisation de l'invention, les dispositifs de routage u_1 à u_7 ont en outre connaissance de leur appartenance au domaine réseau d'un capteur t_1, t_2, t_3 .

5 Les capteurs t_1, t_2, t_3 sont en outre conçus pour exécuter une version distribuée de l'algorithme de Ford-Belman au moyen duquel, à partir d'un ensemble de chemins utilisables, chaque capteur calcule le plus court chemin existant entre lui et un autre capteur destinataire de données à transmettre. Cet algorithme est décrit par exemple dans l'ouvrage intitulé « Graphes et algorithmes », des auteurs M. Gondran et M. Minoux, publié aux Editions Eyrolles.

10 En référence à la figure 2, on décrit ci-dessous un mode de réalisation du procédé communication de données, qui permet d'améliorer la gestion de la bande passante du réseau de capteurs G. Le procédé est mis en œuvre dans chacun des capteurs t_1, t_2, t_3 . En variante, le procédé peut être mis en œuvre dans un seul des capteurs t_1, t_2, t_3 , ou bien dans une partie des capteurs t_1, t_2, t_3 .

15 Le procédé est mis en œuvre par un capteur t et fait appel, pour un nœud t donné, à des informations topologiques représentatives de la topologie du réseau dans l'environnement de ce nœud t , ces informations topologiques comprend notamment l'ensemble des voisins du nœud t , le domaine réseau X_t du nœud t et la frontière de ce domaine réseau. Ces notions sont définies plus en détail ci-dessous.

20 On appelle ensemble X_t un ensemble représentant le domaine du nœud t du sous-ensemble T. L'ensemble X_t comprend des dispositifs du réseau de capteurs G considérés comme appartenant au domaine réseau du capteur t . Le domaine réseau du capteur t peut comprendre, outre le capteur t , un ou plusieurs dispositif(s) de routage u , mais pas d'autre capteur t . Aussi, l'intersection entre l'ensemble X_t et le sous-ensemble T contient uniquement le nœud t ($X_t \cap T = \{t\}$).

25 On appelle ensemble $B(t)$ un ensemble de dispositifs (capteurs t et/ou dispositifs de routage u) considérés comme voisins du domaine X_t et n'appartenant à aucun domaine. L'ensemble $B(t)$ comprend des dispositifs t, u directement reliés à un dispositif du domaine réseau X_t par une liaison, et qui sont considérés comme n'appartenant pas à un autre domaine réseau $X_{t'}$.

30 On appelle ensemble $B'(t)$ une frontière du domaine X_t . L'ensemble $B'(t)$ comprend des dispositifs t, u considérés comme directement reliés par une liaison à un dispositif n'appartenant pas au domaine X_t . L'intersection entre l'ensemble $B'(t)$ et l'ensemble $B'(t')$ est en outre égale à l'ensemble vide ($B'(t) \cap B'(t') = \emptyset$).

On appelle variable X_t^u une variable indicatrice du fait qu'un dispositif de routage u du sous-ensemble $(V-T)$ appartient ou non au domaine X_t du capteur t . La variable X_t^u vaut '1' si le dispositif de routage u appartient au domaine X_t , et vaut '0' sinon.

On appelle ensemble $\Gamma(X)$ un ensemble de dispositifs t, u voisins d'au moins un nœud d'un sous-ensemble X de l'ensemble V ($X \subseteq V$) qui n'appartiennent pas au sous-ensemble X . L'ensemble $\Gamma(X)$ comprend des dispositifs t, u du réseau de capteur G qui n'appartiennent pas au sous-ensemble X mais qui sont directement reliés par une liaison à un dispositif t, u du sous-ensemble X .

On appelle ensemble $E(u - v)$ un ensemble de liaisons formant un chemin de communication d'un premier dispositif u à un deuxième dispositif v du réseau de capteurs G . Un chemin de communication du dispositif u au dispositif v peut comprendre une unique liaison lorsque les dispositifs u et v sont directement reliés par une liaison, ou plusieurs liaisons lorsqu'une communication entre les dispositifs u et v doit être routée via un ou plusieurs dispositif(s) de routage du réseau de capteurs G .

On appelle ensemble $E(X : Y)$ un ensemble de liaisons ayant une extrémité dans le sous-ensemble X de l'ensemble V et l'autre extrémité dans le sous-ensemble Y de l'ensemble V .

On appelle ensemble $pcch(u - v)$ un ensemble de liaisons constituant le plus court chemin (du point de vue du nombre de liaisons) du dispositif u au dispositif v .

On appelle ensemble $P(t - t')$ un ensemble de chemins de communication d'un capteur t à un capteur t' .

L'approche proposée ici consiste à déterminer l'ensemble X_t pour les différents capteurs t du réseau, au moyen d'un processus itératif, comprenant les étapes S21 à S23 décrites ci-dessous, processus au cours duquel chaque capteur détermine son domaine réseau X_t . Le processus est donc mis en œuvre de manière répartie entre les différents capteurs du réseau.

A l'initialisation du processus, chaque ensemble X_t comprenant uniquement le capteur t . Puis, à chaque itération, on ajoute à l'ensemble X_t les voisins de t qui n'appartiennent pas à l'ensemble T et qui n'appartiennent pas déjà à un autre domaine $X_{t'}$, c'est-à-dire qu'on ajoute à l'ensemble X_t le contenu de l'ensemble $B(t)$. Ainsi, à chaque itération, l'étendue de l'ensemble X_t est modifiée à partir de la connaissance de la frontière $B'(t)$ de l'ensemble X_t , à la manière d'un front d'onde qui se propagerait à partir de la frontière de l'ensemble X_t .

Chacun des capteurs t du réseau met en œuvre ce même processus. La variable X_t^u , qui caractérise l'appartenance d'un dispositif de routage u au domaine X_t d'un capteur t , varie ainsi au fur et à mesure des modifications des ensembles X_t déterminés par les capteurs t .

A l'issu du procédé, on aboutit alors à une répartition des entités t, u du réseau dans des domaines réseaux X_t disjoints.

L'algorithme mathématique correspondant est décrit en détail ci-dessous : il fait ressortir les principes énoncés ci-dessus.

5 L'étape S1 est une étape d'initialisation. Pour chaque capteur t du sous-ensemble T , c'est-à-dire dans l'exemple pour chaque capteur t_1, t_2 et t_3 :

- l'ensemble X_t est défini comme comprenant uniquement le capteur t ,
- l'ensemble $B(t)$ est défini comme étant égal à l'ensemble vide, et
- l'ensemble $B'(t)$ est défini comme étant égal à l'ensemble vide.

10 C'est-à-dire que :

Pour tout $t \in T$ $X_t = \{t\}, B(t) = \emptyset, B'(t) = \emptyset$

De plus, pour chaque couple de capteurs t, t' du sous-ensemble T , l'ensemble $P(t-t')$ est défini comme étant égal à l'ensemble vide.

C'est-à-dire que :

Pour tout $t' \in T - \{t\}$ $P(t - t') = \emptyset$

15

De plus, pour chaque dispositif de routage u du sous-ensemble $(V-T)$ la variable X_t^u est fixée égale à '0', c'est-à-dire que les dispositifs de routage u_1 à u_7 sont initialement considérés comme n'appartenant à aucun domaine réseau.

En d'autres termes :

Pour tout $u \in V - T$ $X_t^u = 0$
--

20

L'étape S2 est une étape de traitement itérative visant à établir le plus grand nombre de chemins entre les capteurs t du sous-ensemble T , c'est-à-dire entre les capteurs t_1, t_2 et t_3 .

L'étape S2 comprend des opérations S21 à S25.

A l'opération S21, un capteur t du sous-ensemble T est sélectionné.

25 Puis, un ensemble U_t est déterminé comme contenant les dispositifs de routage u du sous-ensemble $(V-T)$ dont la variable X_t^u est égale à '1'. L'ensemble U_t contient donc les dispositifs de

routage u considérés comme appartenant au domaine réseau X_t du capteur t sélectionné. A l'initialisation du procédé, cet ensemble U_t est donc vide.

L'opération S21 est répétée pour chaque capteur t du sous-ensemble T , c'est-à-dire pour chaque capteur t_1, t_2, t_3 , comme symbolisé par la flèche F_{21} . Lorsque tous les capteurs t ont été
5 sélectionnés, le procédé passe à l'opération S22.

A l'opération S22, un ensemble U est déterminé comme étant égal à l'union des ensembles U_t . L'ensemble U contient donc les dispositifs de routage u considérés comme appartenant au domaine réseau d'un des capteurs t .

A l'opération S23, un capteur t du sous-ensemble T est sélectionné.

10 Puis, l'ensemble $B(t)$ est déterminé comme étant égal à l'intersection de l'ensemble des voisins $\Gamma(X_t)$ et du sous-ensemble $(V-T)$, moins l'union de l'ensemble U et des ensembles X_t associés respectivement à chaque capteur t . L'ensemble $B(t)$ contient donc les dispositifs de routage u , considérés comme voisins du domaine réseau du capteur t , et n'appartenant pas déjà à un domaine réseau d'un autre capteur .

15 Puis, l'ensemble $B'(t)$ est déterminé comme étant égal à l'ensemble $B(t)$, moins l'union des intersections des ensembles $B(t)$ et $B'(t)$ respectivement associés à chaque nœud t du sous-ensemble T . L'ensemble $B'(t)$ contient donc les dispositifs de routage u considérés comme voisins du domaine réseau du capteur t , n'appartenant pas déjà au domaine réseau d'un des capteurs t , et n'appartenant pas à la fois à l'ensemble des voisins d'un des nœuds t et à la frontière du domaine
20 réseau de ce nœud t . La détermination de l'ensemble $B'(t)$ permet de décroiser les frontières des différents domaines réseau.

Puis, l'ensemble $B(t)$ est défini comme étant égal à l'ensemble $B'(t)$.

Puis, le domaine X_t est déterminé comme étant égal à l'union du domaine X_t et de l'ensemble $B(t)$. Le domaine X_t contient donc à ce stade les dispositifs de routage u appartenant
25 déjà au domaine réseau du capteur t , ainsi que les dispositifs de routage u considérés comme voisins du domaine réseau du capteur t et n'appartenant pas déjà à un domaine réseau.

Puis, pour tout dispositif appartenant à l'intersection du domaine X_t et du sous-ensemble $(V-T)$, la variable X_t'' est fixée égale à '1'. Ainsi, la variable X_t'' de chaque dispositif de routage u appartenant au domaine réseau du capteur t est fixée égale à '1', pour indiquer que le dispositif de
30 routage u appartient au domaine réseau du capteur t .

L'opération S23 est répétée pour chaque capteur t du sous-ensemble T , c'est-à-dire pour chaque capteur t_1, t_2, t_3 , comme symbolisé par la flèche F_{23} . Lorsque tous les capteurs t du sous-ensemble T ont été sélectionnés, le procédé passe à l'opération S24.

A ce stade, les ensembles X_t obtenus sont des ensembles disjoints, dans lesquels sont réparties les entités t, u du réseau : chaque capteur t appartient à un ensemble X_t et un seul, et chaque dispositif u appartient à au plus un ensemble X_t .

- 5 Les étapes S24 à S25 décrites ci-dessous définissent un processus itératif pour la détermination de chemins disjoints à partir de ces domaines réseaux disjoints. Dans ce processus de détermination des chemins, comme il s'agit de chemins pour la transmission de données d'un capteur à un autre, chaque chemin aura nécessairement une extrémité dans un domaine réseau d'un capteur et l'autre extrémité dans le domaine réseau d'un autre capteur. En outre, chaque chemin
- 10 aura une première portion dans un premier domaine réseau et une deuxième portion dans un deuxième domaine réseau. On peut donc construire un chemin complet à partir de portions de chemin déterminées domaine par domaine.

L'algorithme mathématique correspondant est décrit en détail ci-dessous et fait ressortir les principes énoncés ci-dessus.

- 15 A l'opération S24, un ensemble $\tilde{T}$ est déterminé comme étant égal au sous-ensemble T .

A l'opération S25, un capteur t du sous-ensemble T est sélectionné. L'opération S25 comprend une sous-opération S251 comportant la sélection d'un capteur t' appartenant à l'ensemble $\tilde{T}$.

- 20 Puis, une liaison (u,v) appartenant à l'ensemble $E(X_t : X_{t'})$ est sélectionnée. Une liaison (u,v) est donc sélectionnée parmi les liaisons ayant une extrémité dans le domaine réseau du capteur t et une extrémité dans le domaine réseau du capteur t' .

- 25 Puis, si les ensembles $pcch(t-u)$ et $pcch(v-t')$ sont non vides, l'ensemble des chemins $P(t-t')$ est déterminé comme étant égal à l'union de l'ensemble $P(t-t')$ et de l'ensemble formé par $pcch(t-u)$, (u,v) et $pcch(v-t')$. Cette opération de détermination est donc réalisée à condition qu'il existe des liaisons permettant de construire un ensemble $(pcch(t-u), (u,v), pcch(v-t'))$, c'est-à-dire à condition qu'il existe, dans l'ensemble E , des liaisons permettant de construire un nouveau chemin de communication entre les capteurs t et t' . Le nouveau chemin de communication est alors formé par le plus court chemin entre le capteur t et le dispositif de routage u , la liaison $(u-v)$ entre le dispositif u et le dispositif v , et le plus court chemin entre le dispositif v et le capteur t' .

- 30 Puis, l'ensemble E est déterminé comme étant égal à l'ensemble E courant moins l'ensemble formé par $pcch(t-u)$, (u,v) , et $pcch(v-t')$. En d'autres termes, les liaisons formant le nouveau chemin de communication sont retirées de l'ensemble des liaisons E , et ne pourront donc plus être utilisées pour construire un nouveau chemin de communication. Cela permet de construire un ensemble de chemins de communication disjoints.

La sous-opération S251 est répétée pour chaque capteur t' de l'ensemble $\tilde{T}$, c'est-à-dire pour chaque capteur t_1, t_2, t_3 , comme symbolisé par la flèche F251. Lorsque tous les capteurs t' de l'ensemble $\tilde{T}$ ont été sélectionnés, le procédé sort de la boucle F251.

5 L'opération S25 est répétée pour chaque capteur t du sous-ensemble T , comme symbolisé par la flèche F25. Lorsque tous les capteurs t du sous-ensemble T ont été sélectionnés, le procédé sort de la boucle F25.

Comme symbolisé par la flèche F2, l'étape S2 est répétée tant qu'il reste un dispositif de routage u du sous-ensemble $(V-T)$ qui n'appartient à aucun domaine X_i . Lorsqu'il ne reste plus de dispositif de routage u du sous-ensemble $(V-T)$ n'appartenant à aucun domaine X_i , le procédé passe
10 à l'étape S3.

L'étape S2 peut être réalisée par l'algorithme ci-dessous :

15

20

25 (suite page suivante)

Tant qu'il existe $u \in V - T$ tel que $X_t^u = 0$

Pour tout $t \in T$

$$U_t = \{u \in V - T, X_t^u = 1\}$$

$$U = \cup_{t \in T} U_t$$

Pour tout $t \in T$

$$B(t) = (\Gamma(X_t) \cap (V - T)) - (U \cup (\cup_{t \in T} X_t))$$

$$B'(t) = B(t) - \cup_{t \in T} (B(t) \cap B'(t))$$

$$B(t) = B'(t)$$

5

$$X_t = X_t \cup B(t)$$

$$\text{Pour tout } u \in X_t \cap (V - T) \quad X_t^u = 1$$

$$\tilde{T} = T$$

Pour tout $t \in T$

Pour tout $t' \in \tilde{T}$

Prendre une arête $(u, v) \in E(X_t : X_{t'})$

Si $pcch(t - u) \neq \emptyset$ et $pcch(v - t') \neq \emptyset$ alors

$$P(t - t') = P(t - t') \cup (pcch(t - u) \cup (u, v) \cup pcch(v - t'))$$

10

$$E = E - (pcch(t - u) \cup (u, v) \cup pcch(v - t'))$$

A l'étape S3, l'ensemble des chemins de communication disjoints déterminé à l'étape S2 est utilisé pour configurer les dispositifs t, u du réseau de capteurs G . En particulier, l'ensemble de chemins de communication est mémorisé dans le capteur t ayant mis en œuvre le procédé.

15

Puis, un message contenant des informations relatives à l'ensemble des chemins disjoints est transmis, via le module d'émission-réception 2, aux dispositifs de routage u du réseau de capteurs G . Ainsi, les dispositifs de routage u ont connaissance des chemins qui peuvent être utilisés pour la communication entre les capteurs t , ce qui leur permet de router les messages transmis à travers le réseau de capteurs G .

20

Dans le cas où tous les capteurs t du réseau de capteurs mettent en œuvre le procédé, il n'est pas nécessaire de transmettre le message d'information aux autres capteurs t du réseau de capteurs, puisque ceux-ci ont déjà connaissance des chemins de communication qui peuvent être

utilisés entre les capteurs t . Dans le cas où seul un capteur t_1 du réseau de capteurs G met en œuvre le procédé, l'étape S3 comprend également une opération de transmission du message d'information depuis le capteur t_1 à destination des capteurs t_2 et t_3 .

5 Le procédé décrit ci-dessus permet de maximiser le nombre total de chemins disjoints qui relient les capteurs t_1, t_2, t_3 entre eux, c'est-à-dire de déterminer le plus grand nombre de chemins disjoints reliant les capteurs.

Le problème qui consiste à déterminer le nombre maximum de chemins disjoints (arêtes disjoints) entre des terminaux formant un sous-ensemble des nœuds d'un réseau est connu sous le nom de problème de Mader. L'invention propose une solution distribuée au problème de Mader.

10 Le procédé permet en outre, avec un algorithme de faible complexité, de donner un résultat approché au problème de Mader, qui est par exemple décrit dans le document de A. Frank « On a theorem of mader. Discrete Mathematics, 101 :49-57, 1992 ». L'utilisation d'un algorithme de faible complexité permet de mettre ce procédé en œuvre dans des capteurs classiques d'un réseau de capteurs.

15 Lorsque le procédé a été exécuté, chaque communication entre deux capteurs t_1, t_2, t_3 du réseau de capteurs est réalisée en utilisant un chemin de communication de l'ensemble de chemins de communication. Ce procédé itératif permet ainsi de maximiser la bande passante totale que pourront utiliser les capteurs t_1, t_2, t_3 pour communiquer, car l'ensemble des chemins maximise le nombre de chemins disjoints et qu'en conséquence le nombre de chemins utilisables à un instant
20 donné est maximal.

L'utilisation de chemins disjoints permet en outre d'augmenter la sûreté de fonctionnement du réseau de capteurs G . En effet, l'utilisation de chemins disjoints permet d'éviter les risques de collision, donc les risques de non-transmission de données.

On décrit ci-dessous un exemple de réalisation des étapes S1 et S2 du procédé, appliqué au
25 réseau de capteurs de la figure 1.

La figure 3 représente les ensembles $X_{t_1}, X_{t_2}, X_{t_3}$ après l'étape S1. L'ensemble X_{t_1} , qui représente le domaine réseau du capteur t_1 , contient uniquement le capteur t_1 . De manière similaire, l'ensemble X_{t_2} , qui représente le domaine réseau du capteur t_2 , contient uniquement le capteur t_2 , et l'ensemble X_{t_3} , qui représente le domaine réseau du capteur t_3 , contient uniquement le capteur t_3 .

30 La figure 3 représente également les ensembles $B(t_1), B(t_2)$ et $B(t_3)$ après une itération de l'opération S23 pour chaque capteur t_1, t_2, t_3 . L'ensemble $B(t_1)$, des dispositifs de routage voisins du capteur t_1 , contient les dispositifs de routage u_1 et u_3 , qui sont directement reliés au capteur t_1 , respectivement par les liaisons (t_1, u_1) et (t_1, u_3) . De manière similaire, l'ensemble $B(t_2)$, des dispositifs de routage voisins du capteur t_2 , contient les dispositifs de routage u_5 et u_7 , qui sont
35 directement reliés au capteur t_2 , respectivement par les liaisons (t_2, u_5) et (t_2, u_7) , et l'ensemble $B(t_3)$,

des dispositifs de routage voisins du capteur t_3 , contient les dispositifs de routage u_1 , u_4 et u_7 , qui sont directement reliés au capteur t_3 , respectivement par les liaisons (t_3, u_1) , (t_3, u_4) et (t_3, u_7) .

La figure 4 représente les ensembles $B'(t_1)$, $B'(t_2)$ et $B'(t_3)$ après une itération de l'opération S23 pour chaque capteur t_1 , t_2 , t_3 . Dans l'exemple représenté, le capteur t_3 a été sélectionné en premier. Ainsi, l'ensemble $B'(t_3)$ est égal à l'ensemble $B(t_3)$ de la figure 3.

Le capteur t_2 a été sélectionné en deuxième. L'ensemble $B'(t_2)$ a donc été déterminé comme étant égal à l'ensemble $B(t_2)$ moins l'ensemble $B(t_3)$. En conséquence, le dispositif de routage u_7 , qui appartenait à l'ensemble $B(t_3)$ au moment de la détermination, a été retiré de l'ensemble $B(t_2)$ pour obtenir l'ensemble $B'(t_2)$.

Le capteur t_1 a été sélectionné en troisième. De manière similaire, l'ensemble $B'(t_1)$ a donc été déterminé comme étant égal à l'ensemble $B(t_1)$ moins l'ensemble $B(t_3)$ et moins l'ensemble $B(t_2)$. En conséquence, le dispositif de routage u_1 , qui appartenait à l'ensemble $B(t_3)$ au moment de la détermination, a été retiré de l'ensemble $B(t_1)$ pour obtenir l'ensemble $B'(t_1)$.

La figure 5 représente les ensembles X_{t_1} , X_{t_2} et X_{t_3} après une itération de l'opération S23 pour chaque capteur t_1 , t_2 , t_3 . L'ensemble X_{t_1} est déterminé comme étant égal à l'union des ensembles X_{t_1} et $B(t_1)$, après que l'ensemble $B(t_1)$ ait été défini comme étant égal à l'ensemble $B'(t_1)$. Ainsi, à ce stade, l'ensemble $X(t_1)$ contient le capteur t_1 qu'il contenait initialement, ainsi que le dispositif de routage u_3 contenu dans l'ensemble $B(t_1)$. De manière similaire, l'ensemble X_{t_2} contient le capteur t_2 et le dispositif de routage u_5 , et l'ensemble X_{t_3} contient le capteur t_3 et les dispositifs de routage u_1 , u_4 et u_7 .

Les figures 5 à 7 représentent le déroulement de deux itérations de l'opération S25.

On considère que le capteur t_3 est sélectionné à l'opération S25, puis que le capteur t_2 et la liaison (u_4, u_5) sont sélectionnés à la sous-opération S251. La sélection de la liaison (u_4, u_5) est symbolisée sur la figure 5 par la mise en gras de cette liaison.

L'ensemble $P(t_3-t_2)$ est alors déterminé comme étant égal à l'union des ensembles $P(t_3-t_2) = \emptyset$ et $((pcch(t_3-u_4)=(t_3, u_4), (u_4, u_5), pcch(u_5-t_2)=(t_2, u_5))$. L'ajout du chemin $((t_3, u_4), (u_4, u_5), (t_2, u_5))$ dans l'ensemble $P(t_3-t_2)$ est symbolisé sur la figure 6 par la représentation des liaisons (t_3, u_4) et (t_2, u_5) en pointillés. Puis, les liaisons (t_3, u_4) , (u_4, u_5) , et (t_2, u_5) sont supprimées de l'ensemble E , comme représenté sur la figure 7, pour ne pas être réutilisées.

Lors de l'itération suivante de la sous-opération S251, on considère que le capteur t_1 et la liaison (t_1, u_1) sont sélectionnés. La sélection de la liaison (t_1, u_1) est symbolisée sur la figure 5 par la mise en gras de cette liaison. L'ensemble $P(t_3, t_1)$ est alors déterminé comme étant égal à l'ensemble $((t_3, u_1), (t_1, u_1))$. L'ajout du chemin $((t_3, u_1), (t_1, u_1))$ dans $P(t_3-t_1)$ est symbolisé sur la

figure 6 par la représentation de la liaison (t_3, u_1) en pointillés. Puis, les liaisons (t_3, u_1) et (t_1, u_1) sont supprimées de l'ensemble E, comme représenté sur la figure 7.

Lors de l'itération suivante de l'opération S25, le capteur t_2 est sélectionné. Puis le capteur t_1 et la liaison (t_1, t_2) sont sélectionnés à la sous-opération S251. La sélection de la liaison (t_1, t_2) est symbolisée sur la figure 5 par la mise en gras de cette liaison. L'ensemble $P(t_2 - t_1)$ est alors déterminé comme étant égal à l'union de l'ensemble $P(t_2 - t_1) = \emptyset$ et de l'ensemble $(pcch(t_2 - t_1) = (t_2, t_1))$. Puis, la liaison (t_2, t_1) est supprimée de l'ensemble E, comme représenté sur la figure 7.

Le procédé continu ensuite de manière similaire, par itération. Les figures 8 à 10 représentent le déroulement de la dernière itération de l'étape S2.

La figure 8 montre que, à ce stade, le domaine X_{t1} comprend le capteur t_1 et le dispositif de routage u_3 , le domaine X_{t2} comprend uniquement le capteur t_2 , et le domaine X_{t3} comprend le capteur t_3 et le dispositif de routage u_7 . En outre, la frontière $B(t_1)$ est égale à l'ensemble vide, la frontière $B(t_2)$ comprend le dispositif de routage u_7 , et la frontière $B(t_3)$ comprend les dispositifs de routage u_4 et u_6 .

La figure 9 montre que, lors d'une itération de l'opération S25, le capteur t_3 a été sélectionné. Puis, le capteur t_1 et la liaison (u_3, u_4) ont été sélectionnés à S251. A ce stade, il n'y a plus, dans l'ensemble E, de liaison entre le capteur t_3 et le dispositif de routage u_4 . Un nouveau chemin entre les capteurs t_3 et t_1 ne peut donc pas être construit.

A l'itération suivante de la sous-opération S251, le capteur t_2 et la liaison (t_2, u_7) sont sélectionnés. Le chemin $((t_3, u_7), (u_7, t_2))$ est alors ajouté dans l'ensemble $P(t_3, t_2)$ des chemins entre les capteurs t_3 et t_2 .

La figure 10 représente l'ensemble final des chemins disjoints. Cet ensemble comprend le chemin (t_1, t_2) entre les capteurs t_1 et t_2 , le chemin $((t_1, u_1), (u_1, t_3))$ entre les capteurs t_1 et t_3 , et les chemins $((t_2, u_5), (u_5, u_4), (u_4, t_3))$ et $((t_2, u_7), (u_7, t_3))$ entre les capteurs t_2 et t_3 .

Ces quatre chemins de communication sont alors mémorisés dans les différents dispositifs t , u du réseau de capteurs G et seront les seuls chemins de communication utilisés pour les communications entre les capteurs t_1 , t_2 et t_3 . La bande passante du réseau de capteurs G est ainsi optimisée, et la sécurité de la transmission est assurée par l'utilisation de chemins de communication disjoints.

Bien entendu, la présente invention ne se limite pas aux formes de réalisation décrites ci-avant à titre d'exemples ; elle s'étend à d'autres variantes.

Par exemple, le procédé peut être mis en œuvre dans des entités autres que des capteurs, par exemple dans des actionneurs, ou autres types de terminaux. Le réseau informatique n'est donc pas nécessairement un réseau de capteurs, mais peut être, par exemple, un réseau IP utilisant un

protocole de type Multiprotocol Label Switching (MPLS), ou encore un réseau ad-hoc, un réseau véhiculaire, un réseau wifi maillé, un réseau de pairs, etc.

En outre, le calcul des chemins peut être réalisé dans une entité physiquement distincte de l'entité communicante qui utilise ce chemin.

REVENDICATIONS

1. Procédé de traitement de données pour la communication de données dans un réseau (G) comprenant un ensemble de dispositifs ($t_1, t_2, t_3, u_1, u_2, u_3, u_4, u_5, u_6, u_7$) formant nœuds dudit réseau et un ensemble de liaisons ($((t_1, u_1), (t_1, u_3), (t_1, t_2), (t_2, u_5), (t_2, u_7), (t_3, u_1), (t_3, u_4), (t_3, u_7), (u_1, u_2), (u_2, u_5), (u_2, u_6), (u_3, u_4), (u_3, u_6), (u_4, u_5), (u_6, u_7))$), chaque liaison reliant deux desdits dispositifs, le procédé comprenant des étapes :
 - détermination d'un ensemble de chemins de communication disjoints entre les dispositifs, chaque chemin de communication comprenant une ou plusieurs liaison(s) de l'ensemble de liaisons,
 - configuration desdits dispositifs pour que chaque communication entre deux desdits dispositifs utilise un chemin de communication dudit ensemble de chemins de communication.
2. Procédé selon la revendication 1, dans lequel l'ensemble des dispositifs du réseau comprend un sous-ensemble (T) de dispositifs émetteurs / destinataires de données et dans lequel l'étape de détermination de l'ensemble de chemins de communication est mise en œuvre par les dispositifs dudit sous-ensemble (T), chaque dispositif dudit sous-ensemble utilisant pour cette détermination des informations représentatives de la topologie du réseau dans l'environnement de ce dispositif,
3. Procédé selon la revendication 2, dans lequel lesdites informations comprennent la définition d'un domaine réseau auquel appartient un dispositif dudit sous-ensemble.
4. Procédé selon l'une quelconque des revendications 1, dans lequel l'ensemble des dispositifs du réseau comprend un sous-ensemble (T) de dispositifs émetteurs / destinataires de données et dans lequel lors de l'étape de détermination, chaque dispositif dudit sous-ensemble (T) détermine un domaine réseau auquel il appartient, les domaines réseaux déterminés étant disjoints et comprenant chacun un seul dispositif dudit sous-ensemble (T), puis on détermine des chemins disjoints ayant une extrémité dans un des domaines réseaux disjoints déterminés et une autre extrémité dans un autre des domaines réseaux disjoints.
5. Procédé selon la revendication 1, caractérisé en ce que l'étape de détermination de l'ensemble de chemins de communication est mise en œuvre dans un desdits

dispositifs (t_1 , t_2 , t_3), l'étape de configuration comportant une opération de transmission par ledit dispositif, à destination d'au moins un autre dispositif du réseau, d'un message contenant des informations relatives à l'ensemble de chemins de communication.

- 5 6. Procédé selon la revendication 1, caractérisé en ce que l'étape de détermination de l'ensemble de chemins de communication comporte une opération de sélection de deux dispositifs parmi lesdits dispositifs du réseau, et une opération de détermination d'un nouveau chemin de communication entre les deux dispositifs sélectionnés.
- 10 7. Procédé selon la revendication 6, caractérisé en ce que l'opération de détermination d'un nouveau chemin de communication entre les deux dispositifs sélectionnés comprend la sélection d'une liaison de l'ensemble de liaisons, et la construction d'un nouveau chemin de communication en utilisant :
 - ladite liaison ((u,v)) sélectionnée,
 - 15 -un plus court chemin ($pcch(t-u)$) entre un desdits deux dispositifs sélectionnés et une première extrémité de ladite liaison, et
 - un plus court chemin ($pcch(v-t')$) entre la deuxième extrémité de ladite liaison et l'autre desdits deux dispositifs sélectionnés.
- 20 8. Programme informatique comportant des instructions pour la mise en œuvre du procédé selon l'une quelconque des revendications 1 à 7 lorsque ce programme est exécuté par un processeur.
- 25 9. Dispositif, conçu pour être utilisé dans un réseau (G) comprenant un ensemble de dispositifs (t_1 , t_2 , t_3 , u_1 , u_2 , u_3 , u_4 , u_5 , u_6 , u_7) formant nœuds dudit réseau et un ensemble de liaisons ((t_1,u_1), (t_1,u_3), (t_1,t_2), (t_2,u_5), (t_2,u_7), (t_3,u_1), (t_3,u_4), (t_3,u_7), (u_1,u_2), (u_2,u_5), (u_2,u_6), (u_3,u_4), (u_3,u_6), (u_4,u_5), (u_6,u_7)), chaque liaison reliant deux desdits dispositifs, ledit dispositif étant configuré pour :
 - déterminer un ensemble de chemins de communication à utiliser entre les dispositifs du réseau (G), chaque chemin de communication comprenant une ou plusieurs liaison(s) de l'ensemble de liaisons,
 - 30 - configurer les dispositifs du réseau (G) pour que chaque communication entre deux desdits dispositifs utilise un chemin de communication dudit ensemble de chemins de communication.
- 35 10. Dispositif selon la revendication 9, caractérisé en ce que le dispositif est un capteur (t_1) configuré pour transmettre des données de mesure, correspondant à une mesure réalisée par le capteur, à au moins un autre capteur dudit réseau,

ledit capteur étant en outre configuré pour que chaque communication entre ledit capteur et un autre capteur dudit réseau utilise au moins un chemin de communication dudit ensemble de chemins de communication.

- 5 11. Réseau (G) comprenant un ensemble de dispositifs ($t_1, t_2, t_3, u_1, u_2, u_3, u_4, u_5, u_6, u_7$) et un ensemble de liaisons $((t_1, u_1), (t_1, u_3), (t_1, t_2), (t_2, u_5), (t_2, u_7), (t_3, u_1), (t_3, u_4), (t_3, u_7), (u_1, u_2), (u_2, u_5), (u_2, u_6), (u_3, u_4), (u_3, u_6), (u_4, u_5), (u_6, u_7))$, chaque liaison reliant deux desdits dispositifs, au moins un desdits dispositifs étant configuré pour déterminer un ensemble de chemins de communication entre les dispositifs, chaque chemin de communication comprenant une ou plusieurs liaison(s) de
10 l'ensemble de liaisons, lesdits dispositifs étant configurés pour que chaque communication entre deux desdits dispositifs utilise un chemin de communication dudit ensemble de chemins de communication.
12. Réseau (G) selon la revendication 11, caractérisée en ce que l'ensemble de dispositifs comprend des capteurs (t_1, t_2, t_3) et des dispositifs de routage ($u_1, u_2, u_3, u_4, u_5, u_6, u_7$).
15
13. Réseau de capteurs selon la revendication 12, caractérisé en ce que ledit au moins un dispositif configuré pour déterminer l'ensemble de chemins de communication est un capteur et est en outre configuré pour transmettre, à destination d'au moins un dispositif de routage, un message contenant des
20 informations relatives à l'ensemble de chemins de communication.
14. Réseau de capteurs selon la revendication 13, caractérisé en ce que ledit au moins un capteur est en outre configuré pour sélectionner deux capteurs parmi lesdits capteurs du réseau, et pour déterminer un nouveau chemin de communication entre les deux capteurs sélectionnés.
- 25 15. Réseau de capteurs selon la revendication 14, caractérisé en ce que ledit au moins un capteur est en outre configuré pour sélectionner une liaison de l'ensemble de liaisons, et pour construire un nouveau chemin de communication à partir de ladite liaison $((u, v))$ sélectionnée, d'un plus court chemin $(pcch(t-u))$ entre un desdits deux capteurs sélectionnés et une première extrémité de ladite
30 liaison, et d'un plus court chemin $(pcch(v-t'))$ entre la deuxième extrémité de ladite liaison et l'autre desdits deux capteurs sélectionnés.

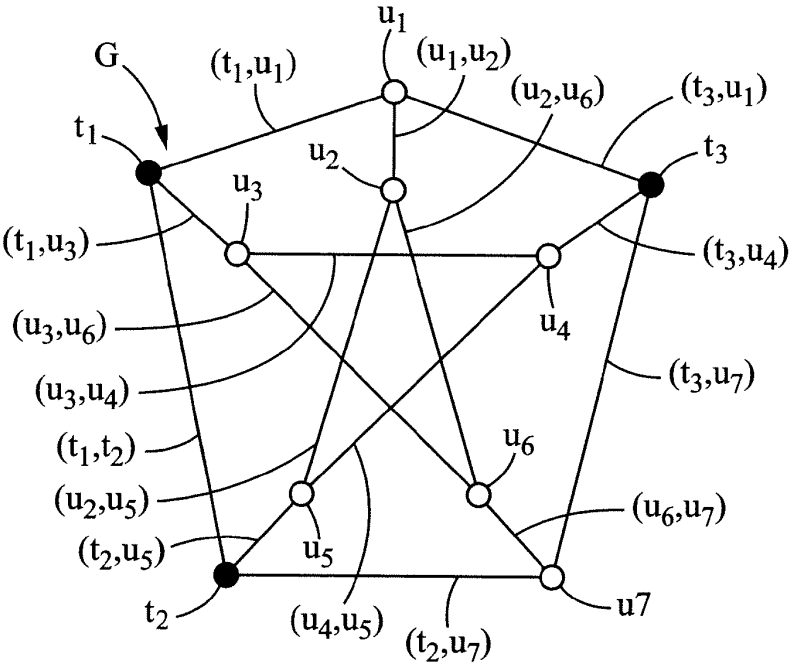


FIG. 1

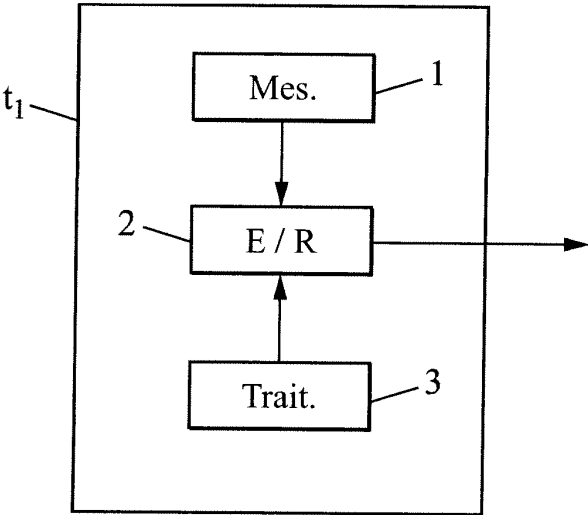


FIG. 11

2/6

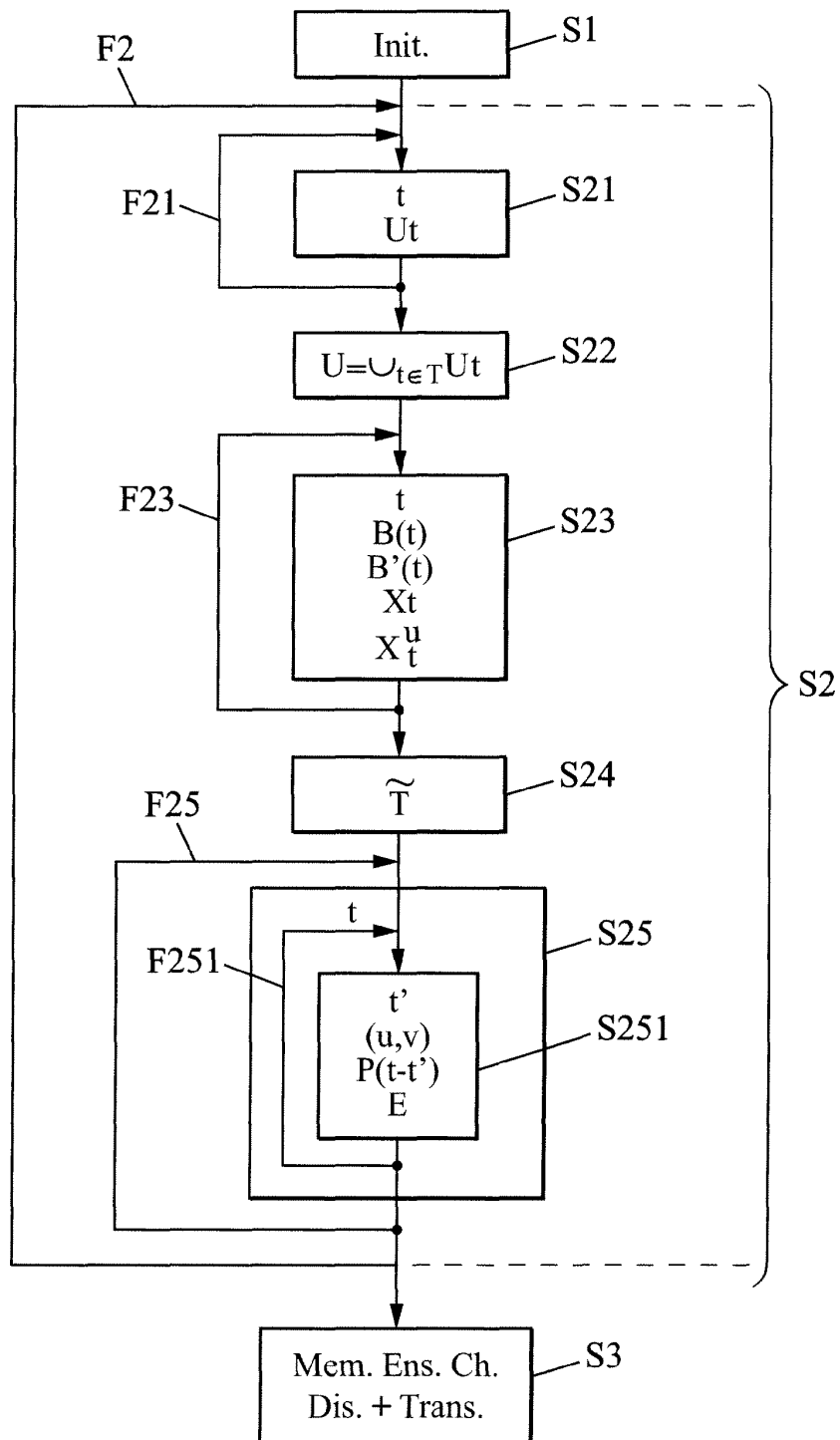


FIG. 2

3/6

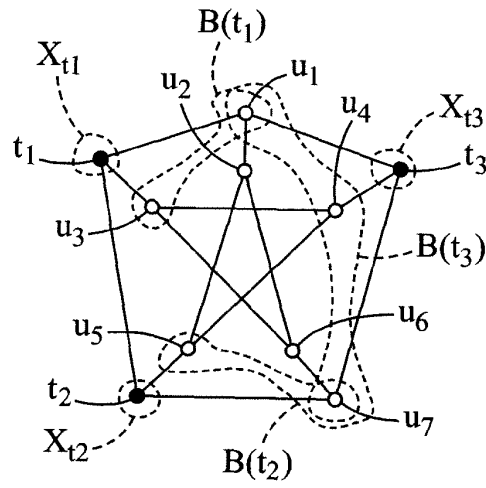


FIG. 3

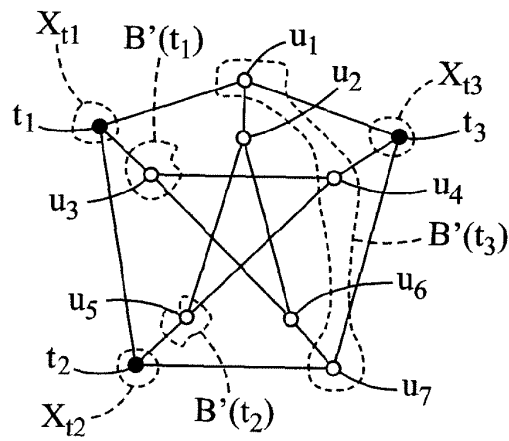


FIG. 4

4/6

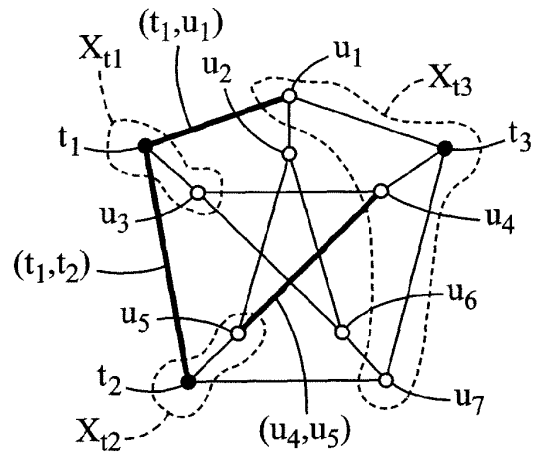


FIG. 5

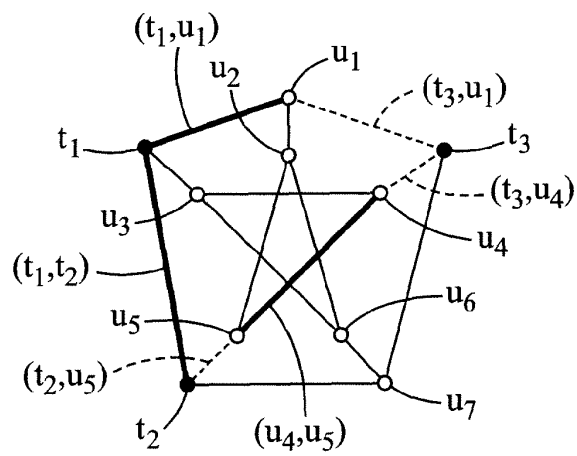


FIG. 6

5/6

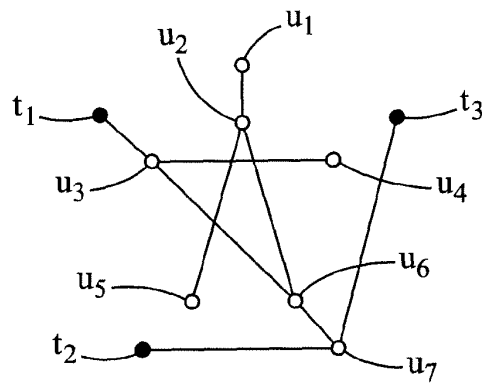


FIG. 7

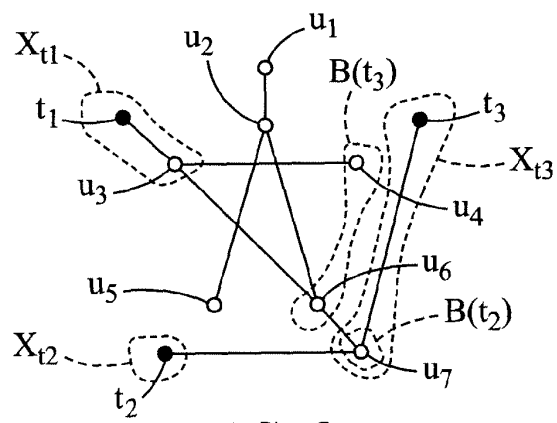


FIG. 8

6/6

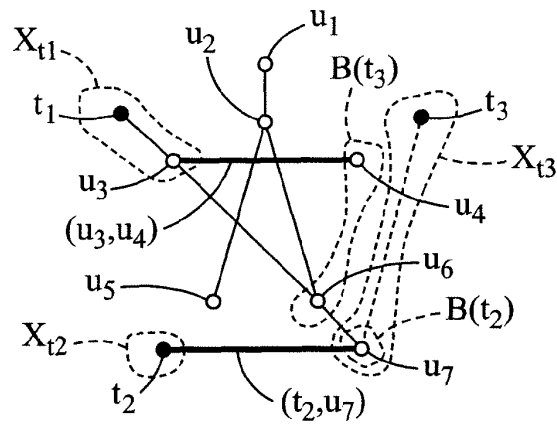


FIG. 9

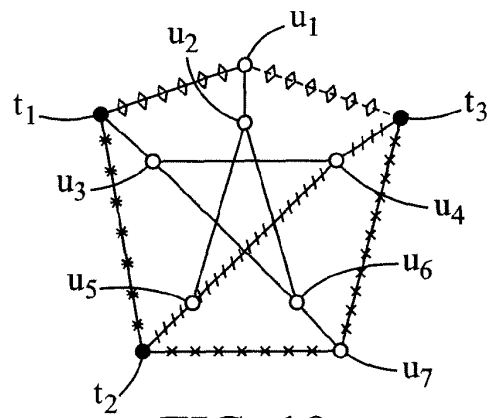


FIG. 10

INTERNATIONAL SEARCH REPORT

International application No

PCT/FR2012/050712

A. CLASSIFICATION OF SUBJECT MATTER

INV. H04L12/56 H04W84/18

ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04L H04W

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2009/228575 A1 (THUBERT PASCAL [FR] ET AL) 10 September 2009 (2009-09-10) paragraph [0001] - paragraph [0002] paragraph [0012] - paragraph [0013] paragraph [0023] - paragraph [0031] page 4, right-hand column, line 4 - line 6 page 5, left-hand column, line 3 - line 7 paragraph [0037] - paragraph [0038] paragraph [0044] paragraph [0049] - paragraph [0050] -----	1-6,8-14
X	US 2005/099983 A1 (NAKAMURA NOBUYUKI [JP] ET AL) 12 May 2005 (2005-05-12) abstract paragraph [0001] - paragraph [0013] paragraph [0056] - paragraph [0057] paragraph [0232] - paragraph [0233] figure 39 ----- -/-	1-6,8-14



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

23 May 2012

Date of mailing of the international search report

31/05/2012

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Martinozzi, A

INTERNATIONAL SEARCH REPORT

International application No

PCT/FR2012/050712

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	<p>MASIP-BRUIN X ET AL: "Research challenges in QoS routing", COMPUTER COMMUNICATIONS, ELSEVIER SCIENCE PUBLISHERS BV, AMSTERDAM, NL, vol. 29, no. 5, 6 March 2006 (2006-03-06), pages 563-581, XP025089784, ISSN: 0140-3664, DOI: DOI:10.1016/J.COMCOM.2005.06.008 [retrieved on 2006-03-06] page 568, right-hand column, line 29 - line 32 paragraph [02.3]</p> <p>-----</p>	1-15

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/FR2012/050712

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2009228575 A1	10-09-2009	NONE	

US 2005099983 A1	12-05-2005	NONE	

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°

PCT/FR2012/050712

A. CLASSEMENT DE L'OBJET DE LA DEMANDE

INV. H04L12/56 H04W84/18

ADD.

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)

H04L H04W

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si cela est réalisable, termes de recherche utilisés)

EPO-Internal

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
X	<p>US 2009/228575 A1 (THUBERT PASCAL [FR] ET AL) 10 septembre 2009 (2009-09-10)</p> <p>alinéa [0001] - alinéa [0002]</p> <p>alinéa [0012] - alinéa [0013]</p> <p>alinéa [0023] - alinéa [0031]</p> <p>page 4, colonne de droite, ligne 4 - ligne 6</p> <p>page 5, colonne de gauche, ligne 3 - ligne 7</p> <p>alinéa [0037] - alinéa [0038]</p> <p>alinéa [0044]</p> <p>alinéa [0049] - alinéa [0050]</p> <p style="text-align: center;">-----</p> <p style="text-align: center;">-/--</p>	1-6,8-14

☒ Voir la suite du cadre C pour la fin de la liste des documents

☒ Les documents de familles de brevets sont indiqués en annexe

* Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent

"E" document antérieur, mais publié à la date de dépôt international ou après cette date

"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)

"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens

"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention

"X" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément

"Y" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier

"&" document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée

23 mai 2012

Date d'expédition du présent rapport de recherche internationale

31/05/2012

Nom et adresse postale de l'administration chargée de la recherche internationale

Office Européen des Brevets, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Fonctionnaire autorisé

Martinozzi, A

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°

PCT/FR2012/050712

C(suite). DOCUMENTS CONSIDERES COMME PERTINENTS		
Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
X	US 2005/099983 A1 (NAKAMURA NOBUYUKI [JP] ET AL) 12 mai 2005 (2005-05-12) abrégé alinéa [0001] - alinéa [0013] alinéa [0056] - alinéa [0057] alinéa [0232] - alinéa [0233] figure 39	1-6,8-14
A	----- MASIP-BRUIN X ET AL: "Research challenges in QoS routing", COMPUTER COMMUNICATIONS, ELSEVIER SCIENCE PUBLISHERS BV, AMSTERDAM, NL, vol. 29, no. 5, 6 mars 2006 (2006-03-06), pages 563-581, XP025089784, ISSN: 0140-3664, DOI: DOI:10.1016/J.COMCOM.2005.06.008 [extrait le 2006-03-06] page 568, colonne de droite, ligne 29 - ligne 32 alinéa [02.3] -----	1-15

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

Demande internationale n°

PCT/FR2012/050712

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
US 2009228575 A1	10-09-2009	AUCUN	
US 2005099983 A1	12-05-2005	AUCUN	

Annexe 2 : Algotel2011 - FTTH

Un algorithme pour le déploiement des réseaux FTTH

Sébastien Imbrosciano¹ [†] and Alexandre Laugier¹ [‡] and Yann Vaxès² [§]

¹ France Telecom/R&D/BIZZ/DIAM 905, rue Albert Einstein, F-06921 Sophia-Antipolis Cedex, France

² LIF - UFR Sciences Luminy, 163 avenue de Luminy, 13288 Marseille Cedex 9, France

Le problème de déploiement optimal d'un réseau FTTH (Fiber To The Home) consiste à concevoir un plan d'installation de fibres optiques de coût minimal permettant de connecter un ensemble de clients à un nœud de raccordement via un ensemble de coupleur. Nous présentons une modélisation de ce problème basée sur le concept de flot avec multiplicateurs et nous montrons comment obtenir par génération de colonnes une solution optimale fractionnaire du programme linéaire correspondant. Cette approche nécessite de résoudre à chaque itération un problème de plus court chemin généralisé multi-contraint. Pour cela, nous décrivons un algorithme pseudo-polynomial qui généralise au cas avec multiplicateurs l'algorithme de plus court chemin multi-contraint introduit dans [DS88].

Keywords: networks, routing, multi-constrained generalized flow, multi-constrained shortest path

1 Introduction

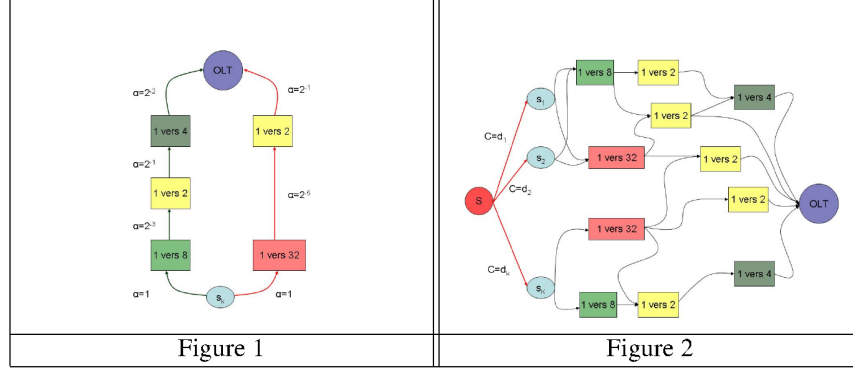
Nous considérons un problème d'optimisation qui apparaît lorsqu'un opérateur de télécommunication souhaite déployer de manière efficace un réseau de fibres optiques pour connecter un ensemble de clients $\mathcal{K}$. Ce problème consiste à déterminer un plan d'installation de fibres optiques permettant de connecter chacun des clients à un nœud de raccordement appelé *OLT* (Optical Line Termination). Pour cela, l'opérateur dispose de sites accueillants des coupleurs qui permettent d'éclater le flux issu d'une fibre optique sur plusieurs fibres. Ici, nous supposons que les coupleurs sont déjà localisés dans le réseau et que leur facteur de concentration de ces derniers est connu. Nous supposons également connus l'ensemble des liaisons qui peuvent être équipées de fibres optiques ainsi que le coût de déploiement d'une fibre sur chacune de ces liaisons. Les facteurs de concentration des coupleurs sont des puissances de 2. Plus précisément, chaque coupleur est caractérisé par un entier $t \in \{1, 2, 3, 5\}$ et permet de réémettre le flux issu d'une fibre sur 2^t fibres. L'architecture réseau impose que sur chaque chemin utilisé pour connecter un client à OLT, le produit des facteurs de concentration soit égal à 64 (voir Fig.1). Finalement, on connaît pour chaque liaison l'affaiblissement subi par le signal optique sur cette liaison et on souhaite également pouvoir garantir que l'affaiblissement subi par le signal sur le chemin entre le client et OLT (i.e. la somme des affaiblissements sur les liaisons du chemin) ne dépasse pas un certain seuil, noté NdB .

Pour modéliser ce problème, nous proposons d'utiliser la notion de flot avec multiplicateurs (voir par exemple [AMO93] pour une définition formelle et des applications). Cette notion se distingue de la notion usuelle de flot par l'existence de multiplicateurs α_e pour chaque arc e qui indique que si une unité de flot entre dans l'arc e alors α_e unités de flot en ressortent. Ces multiplicateurs permettent de prendre en compte la présence des coupleurs en appliquant à tous les flots passant via un coupleur, un multiplicateur égal à l'inverse du facteur de concentration du coupleur (le flot est orienté des clients vers l'OLT, voir Fig.1).

[†]e-mail : Sebastien.Imbrosciano@orange-ftgroup.com

[‡]e-mail : Alexandre.Laugier@orange-ftgroup.com

[§]e-mail : Yann.Vaxes@lif.univ-mrs.fr



2 Modélisation

L'ensemble des liaisons sur lesquels des fibres optiques peuvent être déployées est représenté par un graphe orienté $G = (V, E)$ dont les nœuds sont OLT, les coupleurs, les clients et une super-source S reliée à tous les clients (voir Fig. 2). A chaque arc est assigné une capacité C_e représentant le nombre de fibres que la liaison peut accueillir, un coût d'installation unitaire l_e , un affaiblissement linéique γ_e et un multiplicateur $\alpha_e \leq 1$ qui est égal 1 si l'origine de l'arc e n'est pas un coupleur et à l'inverse du facteur de concentration du coupleur sinon (voir Fig. 1). La super-source S est reliée à chaque client s_k , $k \in \mathcal{K}$ par un arc dont la capacité est égale aux nombres de fibres d_k demandées par le client s_k de coût nul et de multiplicateur unitaire. On note $d := \sum_{k \in \mathcal{K}} d_k$ le nombre total de fibres demandées par les clients et $\mathcal{P}$ l'ensemble des chemins p (considérés comme des listes d'arcs) allant de S à OLT respectant les contraintes technologiques, i.e. (i) le produit des facteurs de concentrations des coupleurs traversés par p est égal à 64 (ii) la somme des affaiblissement sur les arcs du chemin est inférieur ou égal la valeur seuil Ndb . Une unité de flot entrant dans un arc représente le déploiement d'une fibre optique entre l'origine et l'extrémité de cet arc. Pour chaque chemin $p \in \mathcal{P}$, la variable de décision ϕ_p représente la quantité de flot envoyée par S en direction de OLT via le chemin p . Pour chaque arc $e_q \in p := (e_1, e_2, \dots, e_r)$, on note $x_p(e_q) := \prod_{i=1}^{q-1} \alpha_{e_i}$ la quantité de flot qui arrive dans l'arc e_q lorsqu'une unité de flot est émise à partir de S et transite sur les arcs $e_1, e_2, \dots, e_{q-1}$. On remarque que le flot étant porté par des chemins dont le facteur total de concentration est 64, une unité de flot arrivant à OLT correspond exactement à 64 unités de flot émises par S . Nous supposons par la suite que d est divisible par 64. En effet, dans le cas contraire, on peut introduire un client fictif relié à tous les coupleurs par une liaison de coût nul dont la demande complète d pour en faire un multiple de 64.

Le problème *FTTH* revient à calculer une solution optimale entière du problème de flot généralisé multi-constraint suivant.

$$FTTH = \begin{cases} \text{Min} & \sum_{p \in \mathcal{P}} \sum_{e \in p} l_e x_p(e) \phi_p \\ & \sum_{p \in \mathcal{P}} \phi_p = d & : \lambda \\ & \sum_{p \in \mathcal{P}: e \in p} x_p(e) \phi_p \leq C_e, \forall e \in E & : \beta_e \\ & \phi_p \in \mathbb{N}, \forall p \in \mathcal{P} \end{cases} \quad (1)$$

$$(2)$$

La quantité $x_p(e) \phi_p$ représente le nombre de fibres optiques induits sur l'arc e par l'émission à partir de S de ϕ_p unités de flot sur le chemin p . La fonction objectif consiste donc à minimiser le coût total d'installation des fibres optiques. La contrainte (1) assure que chaque client reçoit le nombre de fibres optiques demandées. Les contraintes (2) assurent que le nombre de fibres déployées sur chaque liaison e ne dépasse pas le nombre de fibres qu'elle peut accueillir. Les contraintes technologiques sont prises en compte implicitement dans la définition de l'ensemble $\mathcal{P}$.

Ce modèle introduit une variable entière ϕ_p pour chaque chemin $p \in \mathcal{P}$ qui vérifie les contraintes technologiques. Ce nombre de variables est en général très grand (non-polynomial par rapport à la taille du réseau). La relaxation linéaire associée à ce programme linéaire en nombres entiers est obtenue en remplaçant

chaque contrainte d'intégrité $\phi_p \in \mathbb{N}$ par $\phi_p \geq 0$. Ici, nous nous intéressons uniquement à la résolution optimale de cette relaxation linéaire, nous garderons FTTH comme notation pour ce programme linéaire. Dans la section suivante, nous montrons comment calculer efficacement une solution optimale de cette relaxation en montrant comment identifier un nombre réduit de chemins qui supportent une solution optimale.

3 Résolution par génération de colonnes

Nous allons résoudre une succession de programmes linéaires $FTTH(\overline{\mathcal{P}})$, portant sur les variables associées à un sous-ensemble de chemins $\overline{\mathcal{P}} \subseteq \mathcal{P}$. A chaque itération, la solution optimale duale de $FTTH(\overline{\mathcal{P}})$ permet d'identifier un nouveau chemin dont l'ajout à $\overline{\mathcal{P}}$ permet d'améliorer le plan de déploiement courant. Pour mettre en évidence un critère permettant d'identifier ces chemins, on peut considérer le programme linéaire dual de $FTTH(\overline{\mathcal{P}})$ portant sur les variables duales λ et β_e associées aux contraintes (1) et (2) :

$$FTTH^* = \begin{cases} \text{Max} & \lambda d - \sum_{e \in E} C_e \beta_e \\ & \lambda - \sum_{e \in p} x_p(e) \beta_e \leq \sum_{e \in p} x_p(e) l_e, \quad \forall p \in \mathcal{P} : \phi_p \\ & \beta_e \geq 0, \quad \forall e \in E \end{cases}$$

Soit $(\lambda, \beta_e : e \in E)$ une solution optimale de $FTTH(\overline{\mathcal{P}})$. Trouver un chemin $p \in \mathcal{P} \setminus \overline{\mathcal{P}}$ tel que la solution optimale de $FTTH(\mathcal{P} \cup p)$ soit meilleure que celle $FTTH(\mathcal{P})$ revient à identifier une contrainte du dual $FTTH^*(\mathcal{P})$ violée par la solution duale $(\lambda, \beta_e : e \in E)$, i.e. un chemin $p \in \mathcal{P}$ tel que $\sum_{e \in p} x_p(e)(\beta_e + l_e) < \lambda$.

Ce problème de plus court chemin généralisé multi-contraint revient à trouver parmi l'ensemble des chemins respectant les contraintes technologiques, un plus court chemin généralisé qui permet de faire transiter une unité de flot de S à OLT à moindre coût en prenant $dist(e) := (\beta_e + l_e)$ comme coût de transit d'une unité de flot entrant dans l'arc e . Si le chemin p calculé a un coût $\sum_{e \in p} x_p(e)(\beta_e + l_e)$ strictement inférieur à λ , on l'ajoute à $\mathcal{P}$ et on réitère. Sinon, la solution duale courante $(\lambda, \beta_e : e \in E)$ du programme linéaire $FTTH^*(\overline{\mathcal{P}})$ est également valide pour $FTTH^*(\mathcal{P})$ et donc également optimale pour ce problème. Par dualité, la solution primale courante de $FTTH(\overline{\mathcal{P}})$ est également optimale pour le programme linéaire complet $FTTH = FTTH(\mathcal{P})$ justifiant ainsi la proposition suivante.

Proposition 1 L'algorithme 1 calcule une solution optimale de la relaxation linéaire du problème FTTH.

Données : $G = (V, E)$; $(s_k, d_k) \in V(G) \times \mathbb{R}_+$ pour $k \in \mathcal{K}$; $OLT \in V(G)$; $l \in \mathbb{R}_+^{E(G)}$; $C, 1/\alpha, \gamma \in \mathbb{N}^{E(G)}$
Résultat : $(\phi_p : p \in \mathcal{P})$ une solution optimale de FTTH
 $\overline{\mathcal{P}} \leftarrow$ un ensemble arbitraire de chemins valides;
répéter
 Résoudre le programme linéaire dual $FTTH^*(\overline{\mathcal{P}})$;
 Soit $(\lambda, \beta_e : e \in E)$ une solution optimale $FTTH^*(\overline{\mathcal{P}})$;
 Calculer un plus court chemin généralisé $p \in \mathcal{P}$ avec $dist(e) = (\beta_e + l_e)$ comme coût de l'arc e ;
 si $\sum_{e \in p} x_p(e)(\beta_e + l_e) < \lambda$ **alors**
 $\overline{\mathcal{P}} \leftarrow \overline{\mathcal{P}} \cup \{p\}$;
 fin
jusqu'à $\sum_{e \in p} x_p(e)(\beta_e + l_e) \geq \lambda$;
Renvoyer la solution optimale $(\phi_p : p \in \mathcal{P})$ de $FTTH(\overline{\mathcal{P}})$;

Algorithme 1: Résolution de FTTH par génération de colonnes

4 Plus court chemin généralisé multi-contraint

Le problème du plus court chemin généralisé multi-contraint est NP-Complet, même dans le cas d'une seule contrainte avec tous les multiplicateurs à un [GJ79]. L'algorithme pseudo-polynomial que nous proposons pour résoudre ce problème est basé sur la programmation dynamique au même titre que l'algorithme de label-setting classique décrit dans [DS88] qui résout ce problème lorsque les multiplicateurs sont tous égaux à un. On utilise des *labels* associés à des chemins entre S et un sommet du graphe. Chaque label L est constitué de quatre attributs : un sommet $u(L)$ auquel il fait référence et qui est l'extrémité du chemin

représenté par L , un label $prec(L)$ à partir duquel L a été étendu, une coût $c(L)$, un affaiblissement $\Gamma(L)$ et une concentration $\tau(L)$ qui ensemble caractérisent le chemin en construction.

On dit qu'un label L_1 domine un label L_2 si (i) ils concernent le même sommet $u(L_1) = u(L_2)$, (ii) ils correspondent à des chemins ayant la même concentration $\tau(L_1) = \tau(L_2)$, (iii) le coût et l'affaiblissement du chemin associé à L_1 sont inférieurs ou égaux à ceux du chemin représenté par L_2 , i.e. $c(L_1) \leq c(L_2)$ et $\Gamma(L_1) \leq \Gamma(L_2)$. Dans ce cas, en remplaçant le sous-chemin représenté par L_2 par celui de L_1 dans un chemin entre S et OLT , on préserve le facteur de concentration sans détériorer les autres critères. Par conséquent, il est inutile de conserver le label L_2 .

L'algorithme que nous proposons utilise une file pour stocker l'ensemble des labels. Initialement, cette file contient un seul label L_S dont les attributs sont $u(L_S) = S$, $prec(L_S) = \emptyset$, $\tau(L_S) = 1$, $c(L_S) = 0$, $\Gamma(L_S) = 0$ et qui correspond à un chemin réduit au seul sommet S . Tant que la file n'est pas vide, à chaque itération de l'algorithme, on retire le label L situé en tête de la file. On parcourt l'ensemble des voisins v de $u = u(L)$ et on ajoute à la file le label L' avec $u(L') = v$, $c(L') = (c(L) + dist(uv))/\alpha_{uv}$, $\Gamma(L') = \Gamma(L) + \gamma_{uv}$, $\tau(L') = \tau(L)/\alpha_{uv}$ si les trois conditions suivantes sont vérifiées : (i) L' n'est dominé par aucun autre label L'' tel que $u(L'') = v$, (ii) $\Gamma(L') \leq Ndb$, (iii) $\tau(L') = 64$ si $v(L') = OLT$ et $\tau(L') < 64$ sinon. La mise à jour du coût s'explique par le fait que si une unité de flot en $u(L)$ coûte $c(L)$ alors on peut faire arriver α_{uv} unité de flot en v avec un coût $c(L) + dist(uv)$. En mettant à l'échelle, on obtient une unité de flot en v avec un coût de $(c(L) + dist(uv))/\alpha_{uv}$, voir [FW99] pour une justification plus détaillée. A la fin du processus de marquage, si OLT n'a reçu aucun label, il n'existe aucun chemin respectant les contraintes technologiques entre S et OLT . Dans le cas contraire, le label L^* de OLT dont l'attribut coût est minimum correspond à un plus court chemin généralisé multi-contraint.

La correction de cet algorithme est basée sur les deux observations suivantes. D'une part, si on omet la condition de non-domination l'algorithme considère tous les chemins élémentaires entre S et OLT respectant les contraintes de concentration et d'affaiblissement. D'autre part, ignorer les labels dominés n'élimine aucun chemin optimal. Par ailleurs, chaque sommet reçoit au plus $Ndb \times \log_2 64$ labels non-dominés et chacun de ces labels est considéré un nombre de fois proportionnel au degré sortant du sommet. Cet algorithme s'exécute donc en temps pseudo-polynomial $O(|E(G)| \times 6 \times Ndb)$ justifiant ainsi la proposition suivante.

Proposition 2 *L'algorithme 2 calcule en temps pseudo-polynomial $O(|E(G)| \times 6 \times Ndb)$ un plus court chemin généralisé multi-contraint entre S et OLT .*

Données : $G = (V, E)$; $S, OLT \in V(G)$; $dist \in \mathbb{R}_+^{E(G)}$; $1/\alpha, \gamma \in \mathbb{N}^{E(G)}$
Résultat : Un plus court chemin généralisé multi-contraint de S à OLT
 $Q.put(L_S)$ avec $u(L_S) = s$, $prec(L_S) = \emptyset$, $\tau(L_S) = 1$, $c(L_S) = 0$, $\Gamma(L_S) = 0$;
tant que $Q \neq \emptyset$ **faire**
 $L \leftarrow Q.pop()$;
 pour tous les voisins v **de** $u(L)$ **dont un label** L' **peut être étendu à partir de** L **faire**
 $Q.put(L')$ avec $u(L') = v$, $prec(L') = L$, $c(L') = (c(L) + dist(uv))/\alpha_{uv}$,
 $\Gamma(L') = \Gamma(L) + \gamma_{uv}$ et $\tau(L') = \tau(L)/\alpha_{uv}$;
 $Q.remove(L'')$, pour tout L'' dominé par L' ;
 fin
fin
Renvoyer le chemin associé au label L^* de OLT dont l'attribut coût est minimum.

Algorithme 2: Plus court chemin généralisé multi-contraint de S à OLT

Références

- [AMO93] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows : Theory, Algorithms, and Applications*. Prentice Hall, 1993.
- [DS88] M. Desrochers and F. Soumis. A generalized permanent labeling algorithm for the shortest path problem with time windows. *Information Systems and Operations Research*, 26 :191–212, 1988.
- [FW99] L.K. Fleischer and K.D. Wayne. Fast and simple approximation schemes for generalized flow. *Proceedings of the 10th annual ACM-SIAM Symposium on Discrete Mathematics*, 1999.
- [GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

Résumé

Dans cette thèse, nous étudions plusieurs problèmes d'optimisation issus du domaine des réseaux de télécommunications. Les réseaux étudiés sont les réseaux de fibres optiques et les réseaux de capteurs. Nous nous sommes intéressés à l'analyse et la conception de tels réseaux. Les problématiques étudiées sont : pour les réseaux de fibres optiques, minimiser le coût de déploiement et assurer la qualité de service ; dans les réseaux de capteurs, garantir la sécurité des transmissions et l'énergie consommée par le routage des communications. Pour résoudre ces problèmes nous utilisons des techniques de théorie des graphes, de complexité, de programmation linéaire. Les flots avec multiplicateurs et les chemins avec contraintes de ressources sont aussi abordés. Le premier problème consiste à concevoir un plan d'installation de fibres optiques de coût minimal permettant de connecter un ensemble de clients à un nœud de raccordement via un ensemble de coupleurs en respectant les contraintes technologiques imposées par la norme. Nous proposons une modélisation de ce problème ainsi qu'une méthode de résolution. Le deuxième problème est un problème de flot avec contrainte de délai où le délai pour traverser une liaison est proportionnel à la quantité de flot qui circule sur celle-ci. Nous proposons une preuve de NP-complétude dans le cas général, un algorithme d'approximation facteur 2 dans le cas où le graphe support est un chemin et une heuristique évaluée de façon expérimentale qui calcule en un temps raisonnable de bonnes solutions pour des instances de tailles réelles. Enfin, nous proposons deux protocoles concernant les réseaux de capteurs, qui a donné lieu à deux dépôts de brevets. Le premier, basé sur un algorithme distribué, calcule un ensemble de chemins disjoints entre les terminaux du réseau. Le second maximise la durée de vie d'un réseau de capteurs alimentés par batteries. Des résultats d'expérimentations numériques sont aussi présentés.

Abstract

In this thesis, we study some optimization problem applied to telecommunication networks. We study fiber optical networks and sensor networks. We are interested to analysis and design for these types of networks. The issues studied are : for fiber optic networks, minimize the cost of deployment and ensure quality of service ; for sensors network, ensure the safety of transmissions and the energy consumed. To solve these problems we use techniques as graphs theory, complexity, linear programming, generalized flows and paths with resource constraints. The first problem is to minimize the cost to deploy a fiber optical network which connect a set of customers to a connection node through a set of splitter and deal about technological constraints imposed by the standard. We propose a model and a method of resolution for this problem. The second problem is a flow problem with delay constraint where time to cross a edge is proportional to the amount of flow that flows thereon. We offer a proof of NP-completeness in the general case, an approximation algorithm factor 2 in the case where the support graph is a path and an estimated experimentally an heuristic that calculates good solutions for instances of real sizes. Finally, we propose two protocols for sensor networks, which resulted in two patents. The first, based on a distributed algorithm, calculates a set of disjoint paths between terminals. The second maximizing the lifetime of sensors powered by batteries. The results of numerical experiments are also presented.