LE PROBLEME DE STEINER AVEC COLLECTE DE PRIX : MODELISATION ET RESOLUTION EXACTE

Safa Bhar Layeb

Université Tunis El Manar
Ecole Nationale d'Ingénieurs de Tunis

Thèse présentée en vue de l'obtention du titre de

Docteur en Mathématiques Appliquées

LE PROBLEME DE STEINER AVEC COLLECTE DE PRIX : MODELISATION ET RESOLUTION EXACTE

Elaborée par : Safa Layeb Bhar

Soutenue le 19 février 2009

Devant le jury composé de :

Professeur Mohamed Naceur Ammar, Président
Professeur Mohamed Haouari, Directeur de thèse
Professeur Khaled Mellouli, Rapporteur
Professeur Ridha Mahjoub, Rapporteur
Professeur Salem Mathlouthi, Examinateur

A mes anges gardiens,

SALMA & ALA

Remerciements

Je tiens à remercier Professeur Mohamed Haouari pour ses qualités humaines et scientifiques hors normes. Sa générosité et sa bienveillance ont été de tous les instants. Je tiens à lui exprimer toute ma gratitude.

Je tiens à exprimer toute ma gratitude à Professeur Hanif Sherali pour sa collaboration et ses conseils.

Mes remerciements s'adressent aussi à tous les membres du jury pour leurs conseils et leurs encouragements.

Ma reconnaissance s'adresse à tous ceux qui m'ont aidé dans l'élaboration de ce travail, en particulier tous les membres de l'unité de recherche ROI (Recherche Opérationnelle pour l'Industrie).

Résumé

Dans le cadre de cette thèse, nous nous sommes intéressés à l'étude théorique et la résolution exacte d'un problème NP-difficile d'optimisation combinatoire appelé le problème de Steiner avec collecte de prix ainsi qu'à sa généralisation. Ce problème constitue un cadre unificateur de plusieurs problèmes largement traités dans la littérature, d'où son intérêt théorique. Il est aussi d'un intérêt pratique incontestable vu ses applications dans le secteur des télécommunications.

Nous proposons plusieurs formulations mathématiques du problème dont plusieurs formulations polynomiales compactes. Pour dériver des bornes inférieures, nous avons appliqué la relaxation et la décomposition lagrangienne ainsi que plusieurs approches de résolution de problèmes d'optimisation non différentiable. Une heuristique basée sur les méthodes exactes est proposée. Nous avons aussi développé une procédure exacte de type branch-and-cut. Pour évaluer la performance des différentes approches proposées , nous les avons implémentées en utilisant le langage de programmation Microsoft Visual C++ et le solveur de programmes linéaires CPLEX et nous les avons testées sur des instances de données que nous avons générées aléatoirement et aussi sur des instances benchmark de la bibliothéque *SteinLib*.

Mots clés : problème de Steiner avec collecte de prix, relaxation Lagrangienne, optimisation non-différentiable, génération de coupes, formulation compacte.

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1

Introduction générale

Les technologies de l'information ont connu ces dernières années une évolution fulgurante et ont contraint à une fusion entre les réseaux locaux (Local Area Network, LAN) et les réseaux étendus (Wide Area Network, WAN). Cette convergence du LAN et du WAN exige des solutions de connexion des réseaux surtout avec le volume des données échangées et partagées qui devient de plus en plus important, et le nombre de personnes et systèmes ayant accès à l'information qui se multiplie. Dans le Souci d'optimiser ses installations et de minimiser ses coûts, le secteur des télécommunications s'intéresse de plus en plus à l'optimisation de la conception topologique des réseaux qui sont modélisés, dans la cadre de la théorie des graphes, par des problèmes d'optimisation de structures arborescentes.

Dans ce travail, nous proposons une modélisation généralisée d'une famille de problèmes d'optimisation de structures arborescentes. En effet, nous avons appelé ce cadre unificateur de plusieurs problémes de classe NP-difficile, le problème généralisé de Steiner avec collecte de prix (Generalized Prize Collecting Steiner Tree Problem, GPCSTP). Le GPCSTP est défini comme suit : étant donné un graphe connexe non orienté $G = (V, E)$ où $V = \{0, 1, .., n\}$ est l'ensemble des noeuds, E est l'ensemble des arêtes et 0 est un noeud particulier appelé racine. Chaque noeud $j \in V^* = V \backslash \{0\}$ est muni de deux entiers positifs : un profit p_j et une pénalité γ_j. Chaque arête $e \in E$ est munie d'un poids positif c_e. L'ensemble V est partitionné en K sous-ensembles non vides $V_1, V_2, ..., V_K$. A chaque sous-ensemble V_k $(k = 1, ..., K)$ est associé un quota Q_k. Le GPCSTP consiste à déterminer $V_T \subseteq V$ tel que la somme du poids de l'arbre de poids minimal couvrant V_T et de racine 0 dans

le graphe G et des pénalités des noeuds n'appartenant pas à l'arbre, soit minimum. En plus, il faut que la contrainte suivante soit satisfaite : pour tout sous-ensemble $V_T \cap V_k$ $(k = 1, ..., K)$, la somme des profits des sommets couverts par l'arbre soit au moins égale au quota Q_k. *Le cas particulier correspondant à* $K = 1$ *est appelé le problème de Steiner avec collecte de prix (Prize Collecting Steiner Tree Problem, PCSTP).*

Ce travail de thèse a pour objectif l'étude théorique ainsi que le développement de bornes inférieures pour le GPCSTP et des méthodes exactes pour la résolution du PCSTP. Le plan de cette thèse est organisée comme suit:

Dans le **chapitre 2,** nous rappelons les définitions de la théorie des graphes utiles pour la suite de notre exposé et nous donnons un aperçu rapide et informel de la théorie de la complexité algorithmique. Une étude bibliographique des principaux problèmes d'optimisation de structures arborescentes a été effectuée. A travers ses variantes, le problème de Steiner est particulièrement détaillé puisqu'il s'agit d'un problème de base de plusieurs problèmes d'optimisation des réseaux. Nous consacrons la dernière section de ce chapitre à la présentation du GPCSTP, ses variantes ainsi que ses intérêts théoriques et pratiques. En particulier, sa relation avec les problèmes d'optimisation d'arbres connus de la littérature.

Le **chapitre 3** est dédié au GPCSTP. Une nouvelle formulation mathématique a été proposée ainsi que quatre différentes relaxations Lagrangiennes. Une analyse théorique des bornes inférieures correspondantes a été développée ainsi qu'une étude comparative de plusieurs approches de résolution de problèmes d'optimisation non différentiable.

Dans le **chapitre 4,** nous nous intéréssons au PCSTP. Dans ce chapitre, une méthode approchée basée sur la décomposition a été développé sous deux versions. Cette heuristique décompose le problème PCSTP en deux sous problèmes : un problème d'arbre de poids minimal et un PCSTP défini sur un arbre qu'on a prouvé NP-difficile. Chacun de ces deux problèmes est résolu par un algorithme exact. Nous présentons une étude du pire cas de l'heuristique proposée ainsi qu'une étude expérimentale de sa performance.

Dans le **chapitre 5,** une deuxième formulation mathématique basée sur les coupes est développée à laquelle on a appliqué un algorithme exact de génération de coupes. Un prétraitement est mis en place pour réduire la taille

du problème. Le graphe est réduit grâce à deux techniques : une réduction basée sur la relaxation lagrangienne et une réduction basée sur la relaxation linéaire. Une procédure exacte de type branch-and-cut est présentée pour le PCSTP avec une étude expérimentale testant ses résultats.

Le **chapitre 6** est consacré au développement de trois familles de formulations *polynomiales compactes* pour le PCSTP. Une étude théorique comparative des relaxations linéaires correspondantes a été ccomplie. Enfin, une étude expérimentale comparant les relaxations linéaires des formulations proposées ainsi que la résolution exacte et approchée correspondante est présentée.

Nous terminons ce mémoire de thèse par une conclusion générale qui résume les contributions proposées ainsi que plusieurs directions de recherche qui pourront donner suite à ce travail.

Chapitre 2

Les problèmes d'optimisation des structures arborescentes: Un tour d'horizon

2.1 Introduction

Si la notion de structure arborescente n'a été formalisée, en termes mathématiques, que tardivement (au $XIX^{ième}$ siècle), la présentation arborescente d'un exemple de données est une pratique bien antérieure. On peut par exemple citer les arbres généalogiques, mais aussi la représentation de vertus par des arbres de la sagesse dans certains livres médiévaux, ou encore les arbres de classification d'espèces animales de naturalistes des $XVII^{ième}$ et $XVIII^{ième}$ siècles [208].

La première formalisation mathématique du concept d'arbre est généralement attribuée à Kirchhoff dans les années 1840. Elle est issue de travaux portant sur l'étude des circuits éléctriques. C'est cependant Cayley qui a introduit le terme d'arbre, dans une série d'articles (le premier datant de 1857), portant principalement sur l'énumération d'arbres et d'arborescences en relation avec l'étude des formules algébriques [24].

A la suite de ces premiers travaux de Cayley, l'étude des propriétés des structures arborescentes s'est rapidement imposée comme un domaine classique et fécond de la combinatoire, notamment de la combinatoire énumérative [28].

Cet intérêt pour les structures arborescentes a ensuite connu un nouvel essor avec le développement de l'informatique à travers les travaux de Knuth [129].

Dans ce chapitre, nous exposons quelques notions de base de l'optimisation combinatoire et nous nous intéréssons aux incontournables problèmes sur les structures arborescentes.

Vu son importance historique, nous commençons par introduire le problème de l'arbre couvrant de poids minimal. Ensuite, nous exposons les problèmes d'arbre les plus connus de la littérature à savoir le problème de l'arbre de cardinalité k, le problème de l'arbre couvrant avec contrainte de capacité, le problème d'arbre généralisé de poids minimal et le problème de Steiner. Comme le problème de Steiner a de très nombreuses applications, allant du réseau d'interconnexions de longueur minimale entre les différents éléments d'un circuit VLSI, à la construction d'arbres phylogéniques en bioinformatique, ce problème et ses généralisations sont étudiés de façon approfondie.

La dernière section de ce chapitre est consacré au problème de Steiner avec collecte de prix généralisé (Generalized Prize Collecting Steiner Tree Problem, GPCSTP) à travers une nouvelle modèlisation d'optimisation d'arbre en terme de théorie de graphe. Nous présentons les intérêts pratiques et théoriques du GPCSTP qui ont motivés ce présent travail.

2.2 Notions de base

Nous présentons ici le vocabulaire utilisé dans la modélisation des problèmes de réseaux et dont un lecteur habitué peut se dispenser et passer directement au paragraphe 2.3.

2.2.1 Théorie des graphes

La théorie des graphes est devenue un outil privilégié de modélisation et de résolution de nombreux problèmes traitant des structures combinatoires complexes. L'ensemble des techniques et outils mathématiques mis au point permettent de démontrer facilement des propriétés, d'en déduire des méthodes de résolution, des algorithmes,...etc. En effet, les graphes permettent de manipuler plus facilement des objets et leurs relations avec une représentation graphique naturelle: il s'agit d'une abstraction de la réalité de sorte à permettre sa modélisation.

Ainsi, les graphes peuvent servir à modéliser de nombreuses applications à savoir la représentation de connexions et de possibilités de cheminement: détermination de la connexité d'un réseau quelconque (électrique, d'adduction d'eau), calcul du plus court chemin dans un réseau routier. Il sont aussi précieux pour décrire des systèmes dynamiques, c'est à dire évoluant dans le temps: diagrammes de transition, automates d'états finis, chaînes de Markov [176].

Au niveau de cette partie, nous allons introduire quelques définitions de base de la théorie des graphes qui seront utilisées tout au long de ce rapport.

Definition 1 *Un graphe G est un couple (V, E) où V est un ensemble fini de sommets ou de noeuds $V = \{1, ..., n\}$ et $E = \{e_1, ..., e_m\}$ un ensemble d'arêtes avec l'arête $e_k = \{i, j\}$; $i, j \in V$.*

Soit $G = (V, E)$ un graphe non orienté à n sommets et m arcs. A chaque arc $e = (i, j) \in E$ est associé un nombre réel c_{ij} appelé poids ou longueur de l'arc (i, j). Ainsi, on définit la matrice des coûts $C \equiv (c_{ij})_{(i,j)\in E}$.

Definition 2 *Un* arbre *du graphe $G = (V, E)$ est un graphe partiel connexe et sans cycle.*

Dans un graphe connexe, un arbre connecte tous les sommets et comporte $n - 1$ arêtes si le graphe comporte n sommets.

Definition 3 *On définit le poids d'un arbre comme étant la somme algébrique des coûts des arêtes qui le constituent.*

Definition 4 *Une* forêt *de G est un graphe partiel sans cycle.*

Definition 5 *Une* arborescence *est un "arbre orienté": tous les sommets sont descendants d'un sommet unique r appelé racine de G (voir figure 2.1).*

Les arbres ont une grande importance dans les problèmes d'optimisation de réseaux et ce pour les raisons suivantes:

- Un arbre connecte n'importe quel ensemble de noeuds permettant ainsi la connexité du réseau.

- Les problèmes d'optimisation d'arbres se manifestent de manière étonnante dans plusieurs domaines d'application tels que les réseaux informatiques, la distribution d'énergie, ...

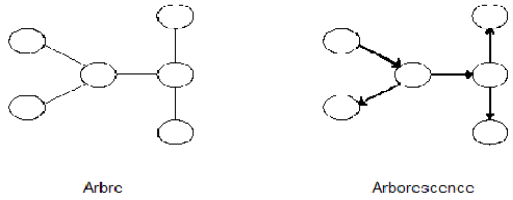

Figure 2.1: Exemple d'arbre et d'arboresence

2.2.2 Complexité des algorithmes

La théorie de la complexité s'attache à connaitre la difficulté, ou la complexité, d'une réponse par un algorithme à un problème donné. La complexité d'un algorithme est une évaluation du nombre d'instructions et de la place mémoire utilisée pour l'exécuter.

Classe P Un problème de décision est dans P s'il peut être décidé sur une machine déterministe en temps polynômial par rapport à la taille n des données. On qualifie alors le problème de polynômial, c'est un problème de complexité $O(n^k)$ pour un certain réel k.

Classe NP La classe NP possède une définition moins naturelle que celle de P, et son nom est trompeur : il ne signifie pas "non polynômial". En effet, la classe NP se compose des problèmes Non-déterministes Polynomiaux et elle réunit les problèmes de décision qui peuvent être décidés sur une machine non déterministe en temps polynômial. De façon équivalente, c'est la classe des problèmes qui admettent un algorithme polynômial capable de tester la validité d'une solution du problème, on dit aussi capable de construire un certificat. Intuitivement, les problèmes dans NP sont les problèmes qui peuvent être résolus en énumérant l'ensemble des solutions possibles et en les testant à l'aide d'un algorithme polynômial.

Conjecture : $P \subset NP$ Comme un problème polynômial peut se résoudre en engendrant une solution et en la testant à l'aide d'un algorithme polynômial, alors tout problème dans P est ainsi dans NP.

En revanche, la réciproque : $NP \subset P$, qui est la véritable difficulté de l'égalité P = NP, est l'un des problèmes ouverts les plus fondamentaux et

intéressants en informatique théorique. Il a été posé en 1970 indépendamment par Stephen Cook et Leonid Levin. La plupart des spécialistes conjecturent que les problèmes NP-difficiles ne peuvent pas résolus en un temps polynômial, et parsuite, $P \neq NP$.

Algorithmes d'approximation Depuis trois décénnies, les chercheurs essaient de concevoir des algorithmes d'approximation qui fournissent, en un temps polynômial, des solutions dont l'écart à l'optimum peut être borné supérieurement.

Definition 6 Un algorithme de ρ-approximation *est un algorithme polynômial qui fournit une solution approchée garantie au pire des cas à un facteur ρ de l'optimal* ($\rho > 1$).

Un tel algorithme est aussi appelé un algorithme d'approximation de facteur ρ.

Autrement dit, pour toute instance I d'un problème de minimisation admettant *un algorithme de ρ-approximation,* si on note Z_I^* la solution optimale de I et UB_I la solution retournée par l'algorithme, on a:

$$Z_I^* \leq UB \leq \rho Z_I^* \quad \forall I \text{ instance}$$

Dans la section suivante, nous allons présenter le problème d'arbre couvrant de poids minimal, considéré comme un modèle de base de plusieurs problèmes d'optimisation basés sur les structures arborescentes.

2.3 Le problème de l'arbre couvrant de poids minimal

En 1926, Otakar Borůvka (1899-1995) un ingénieur et mathématicien tchèque a été le premier à consiédérer le problème de l'arbre couvrant de poids minimal (Minimum Spanning Tree, MST) qu'il a traité dans deux papiers publiés en langue tchèque. Ces papiers historiques étaient plus tard traduits en anglais par Nešetřil et.al [163] et dont un extrait est le suivant:

There are n points in the plane whose mutual distances are different. The problem is to join them with a net in such a way that:

1. any two points are joined to each other either directly or by means of some other points;

2. the total length of the net will be minimal.

Dans sa définiton, Borůvka a supposé que les coûts des arêtes sont différents deux à deux. Il a était motivé par l'application de ce problème dans la conception du réseaux électrique en Moravie qui est aujourd'hui la partie orientale de la république tchèque. Il a dévéloppé un algorithme qui porte son nom avec lequel il a réussit à résoudre une instance de données réelles avec 40 villes.

Pour la fascinante histoire de ce mathématicien exceptionnel ainsi que celui de ce problème nous citons le papier de Nešetřil et. al [163] et aussi celui de Graham et Hell [91].

Nous présentons dans la suite la définition exacte de ce problème en terme de théorie de graphes.

Definition 7 *Soit $G = (V, E)$ un graphe connexe. V est l'ensemble des sommets et E est l'ensemble des arêtes. A chaque arête $e \in E$ est associé un nombre réel c_e, appelé poids ou longueur de l'arête e.*

Le problème de l'arbre couvrant de poids minimal ou simplement arbre de poids minimal (Minimum Spanning Tree MST), consiste à déterminer un arbre de G de poids total minimal.

Ce problème a été largement traité dans la littérature [176]. Le problème de l'arbre couvrant de poids minimal se pose dans plusieurs problèmes réels. Nous citons à titre illustratif l'application suivante: On désire relier n villes par un réseau routier de coût minimum. Les sommets du graphe représentent les villes, les arêtes représentent les trançons qu'il est possible de construire, et les poids des arêtes correspondent aux coûts de construction des trançons.

Algorithmes de résolution

L'intérêt des applications utilisant la structure d'arbre a très tôt motivé la recherche d'algorithmes efficaces. En effet, le problème de l'arbre couvrant de poids minimal est connu depuis les années 50 et les deux premiers algorithmes efficaces qui sont apparus sont ceux de Kruskal et de Prim [176].

Algorithme de Kruskal

En 1956, Kruskal était le premier à proposer une méthode exacte de recherche d'un arbre couvrant de poids minimal sur un graphe non orienté [134]. L'approche de son algorithme consiste en la construction d'un sous-graphe du graphe initial par ajout successif des arêtes. A chaque itération, une nouvelle arête qui ne crée pas de cycle, et dont le poids est minimum, est ajoutée à la liste. Lorsqu'on séléctionné $n-1$ arêtes du graphe initial contenant n sommets et m arêtes, le sous-graphe obtenu constitue un arbre couvrant de poids minimal.

Le principe de l'algoritme de Kruskal est le suivant:

Partant d'un graphe $G = (V, E)$, notons $T = (V, E_T)$ l'arbre recherché.

Initialisation

$E_T = \{\}$

Trier les arêtes E par ordre de poids croissant

Tant que $|E_T| < n - 1$ faire

Séléctionner la première arête a de E

$E = E \backslash \{a\}$

Si l'ajout de l'arête a à l'ensemble E_T ne forme pas un cycle alors $E_T = E_T + \{a\}$

Fin tant que

L'algorithme converge en au moins $n-1$ itérations, mais ceci peut prendre jusqu'à m itérations, si seule la dernière arête permet de compléter l'arbre. Une implémentation efficace de l'algorithme de Kruskal donne une complexité globale de $O(m + n \log n)$ [176].

Algorithme de Prim

Peu de temps aprés Kruskal, en 1957, Prim publie un article [175] où il présente une autre méthode de construction de l'arbre couvrant de poids minimal. En utilisant l'algorithme de Prim, l'arbre couvrant de poids minimal est construit de la façon suivante: on part d'un arbre initial réduit à un seul sommet. Ensuite, à chaque itération, on augmente l'arbre en le connectant au noeud qui lui est relié par l'arête de poids le plus faible.

Le principe de l'algorithme de Prim est le suivant:

Partant d'un graphe $G = (V, E)$, notons $T = (V, E_T)$ l'arbre recherché.

Initialisation

$E_T = \{\}$

Choisir un sommet quelconque de v_0 de V

$V_T = \{v_0\}$ avec V_T est l'ensemble des noeuds ajoutés

Tant que $V_T \neq V$

Sélectionner les arêtes incidentes de chaque arête de V_T

Choisir l'arête de coût minimal, soit v_f le noeud extrémité de l'arête sélectionnée.

$V_T = V_T + \{v_f\}$

Fin tant que

En $n - 1$ itérations, on forme un arbre couvrant de poids minimal. Une bonne implémentation de l'algorithme de Prim admet une complexité de $O(n^2)$ [176].

Les algorithmes de Kruskal et de Prim sont les algorithmes les plus classiques et les plus populaires pour résoudre le MST. Durant les deux dernières décénnies, des algorithmes plus sophistiqués ont été développés, nous citons en particulier ceux de Fredman et Tarjan [72] et Karger et al. [122]. Ce dernier a été présenté en 1995 comme le premier algorithme de résolution du MST avec un temps linéaire en fonction de la taille des instances. Une description détaillée ainsi qu'une étude comparative de ces algorithmes sont présentées dans le papier de Bazlamaçci et Hindi [14]. Ils ont comparé la performance des différents algorithmes développés pour résoudre le MST sur des instances générées aléatoirement dont la taille atteint 16000 noeuds et 524 000 arêtes. Ils ont conclu que les algorithmes classiques (Prim et Kruskal) restent les plus performants sur les instances de taille raisonable.

Pour un revue de la littérature plus dévéloppée sur l'historique de ce problème nous citons les travaux de Pierce [168], Maffioli [150] et Graham et Hell [91].

Le problème de l'arbre couvrant de poids minimal est donc facile à résoudre puisque les algorithmes de Kruskal et de Prim sont polynomiaux. D'autres problèmes d'optimisation d'arbre tiennent compte de contraintes supplémentaires dans la recherche de l'arbre de poids minimal. La considération de ces contraintes rend ces problèmes de classe NP-difficile, d'où le grand intérêt théorique et pratique de leur étude.

2.4 Le problème de l'arbre de cardinalité k

Definition 8 *Etant donné un graphe $G = (V, E)$ et un entier $k \leq |V|-1$. Le problème de l'arbre de cardinalité k (le k-MST) consiste à trouver un arbre de poids minimal dans G qui connecte exactement k arêtes. On l'appelle aussi le problème du k-arbre.*

Il est clair que si $k = |V| - 1$, le k-MST sera réduit au problème de l'arbre couvrant de poids minimal. Contrairement au problème de l'arbre couvrant de poids minimal, le problème de l'arbre de cardinalité k est NP-difficile [67].

En 1994, Fischetti et al. ont établi une étude polyhedrale de ce problème [67]. Quatre ans plus tard, un agorithme exacte de branch-and-cut a été décrit par Ehrgott and Freitag [56] pour résoudre le k-MST. Concernant le dévéloppement de bornes inférieures, Kataoka and Araki [125] ont proposé une méthode qui est efficace uniquement lorsque k est petit.

Exemples d'applications

Le k-MST a été décrit pour la première fois par Fischetti et al. [67] comme une modélisation du problème rencontré par les compagnies pétrolières. En effet, une étude du terrain d'un gisement pétrolier révèle n puits potentiels (noeuds) ainsi que les pipelines possibles connectant les différents puits (arêtes). A cause des restrictions budgétaires, il faut choisir à exploiter seulement k puits parmis les n puits possibles et aussi de les connecter au moindre coût.

Le problème de l'arbre de cardinalité k présente plusieurs d'autres applications telque les plans d'aménagement [71], les réseaux de télécommunication [77] et le partitionnement des graphes [21].

Heuristiques et métaheuristiques

Depuis les années 90, plusieurs heuristiques ont été proposées pour ce problème.

Une première heuristique s'inspire du principe de l'algorithme de Prim. L'algorithme dévéloppé est appelé l'algorithme du K-Prim [57]. Cet algorithme construit n sous-arbres de cardinalité k en commençant à chaque fois par un sommet s du graphe G selon l'algorithme de Prim. En effet, il suffit d'ajouter les arêtes ayant le plus petit poids à la solution courante, initialisée à $\{s\}$, jusqu'à ce qu'on couvre $k-1$ arêtes. Parmis les n sous-arbres ainsi construits, on choisit comme solution heuristique celui qui a le plus petit poids.

Une deuxième heuristique consiste à appliquer l'algorithme Dual Greedy [57]. On commence par considérer tout le graphe G. On supprime les arêtes de poids le plus grand possible de manière greedy (au fur et à mesure) jusqu'à ce qu'on forme un arbre avec $k-1$ arêtes. Il faut aussi vérifier que le graphe reste connexe au cours de la supression des arêtes. Contrairement à l'algorithme du K-Prim, Erghot et al. [57] ont trouvé que l'algorithme DualGreedy donne, dans la plupart des cas, des solutions assez proches de l'optimum.

En plus, et vue sa complexité, plusieurs méta-heuristiques ont été proposées pour le k-MST. Le principe d'une recherche locale a été proposé par Blum et Ehrgott [20]. Une heuristique basée sur la recherche tabou a été testée par Jornstern et Lokketangen [119]. D'autres métaheuristiques ont été appliquées au k-MST telles que les algorithmes génétiques et l'optimisation par les colonies de fourmis [18]. Récement, Blum et Blesa [19] ont appliqué des nouvelles métaheuristiques hybrides sur le problème du k-MST.

En 1996, Garg [76] a présenté un algorithme d'approximation de facteur 3 pour le k-MST. Une modification de cet algorithm a été proposée par Arya et Ramesh [6] permettant amélioration du facteur d'approximation de 2.5. Récemment, Garg [75] a affiné l'algorithme permettant une approximation du problème avec un factor égale à 2. En 2004, Chudak et al. [32] ont montré que l'algorithme de Garg, présenté en 1996, peut être interprété comme une application de la technique de relaxation lagrangienne et de la méthode primal-dual des algorithmes d'approximation.

2.5 Le problème de l'arbre couvrant avec contrainte de capacité

Definition 9 *Soit un graphe non orienté $G = (V, E)$, où $V = \{0, 1, ..., n\}$ est l'ensemble des sommets et E est l'ensemble des arêtes. Le noeud 0 est un noeud central particulier qu'on appelle le noeud racine . Un coût positifs c_e est associé à chaque arête $e \in E$ et une demande positive p_j est associé à chaque noeud $j \in V^* = V \backslash \{0\}$.*

Etant donné un entier Q qu'on appelle capacité, le problème d'arbre couvrant de poids minimal avec contrainte de capacité (Capacitated Minimum Spannning Tree $CMST$) consiste à trouver un arbre couvrant de poids minimal tel que la somme des demandes des noeuds de chaque composante connexe, incidente au noeud racine, ne dépasse pas la capacité Q.

Le CMST a été introduit par Chandy et Lo [25] en 1973. Ils ont proposé de le relaxer en un problème d'arbre couvrant de poids minimimal (MST) pour obtenir une borne inférieure. En 1978, Papadimitriou [166] a montré que le CMST est $NP - complet$ pour $3 \leq Q \leq \frac{|V|}{2}$. Le papier d'Amberg et al. [3] représente une excellente référence pour le CSTP.

Le CMST est un problème de base dans la conception de réseau de télécommunication où on cherche à concevoir un réseau centralisé. On se donne un processeur central et un ensemble de terminaux ayant des demandes spécifiques représentant le traffic des données qui doivent circuler entre le processeur central et les terminaux. Il faut donc construire un réseau pour transmettre ces demandes. Entre toute paire de terminaux et entre le processus central et les terminaux, il y a des liens potentiels dont les coûts peuvent être évalués . Le problème consiste à trouver un réseau connectant les terminaux au processeur central en respectant les trois propriétés suivantes:

1- Le réseau est un arbre (sans cuircuit).

2- La somme des demandes de chaque sous-arbre connecté au noeud racine ne doit pas dépasser une quantité fixe Q.

3- Le coût total de la connexion du réseau est le plus faible possible.

Vues ses applications multiples, un grand intérêt a été porté au CMST et plusieurs formulations mathématiques ont été proposées. Nous citons en particulier celles proposées par Gavish ([79], [78]) et Gouveia ([88], [86]).

Parmis les méthodes exactes dévéloppées pour résoudre ce problème, on cite la méthode de génération de coupes appliquée par Hall [96] et Gouveia et Martins [89] et dont le défaut majeur est le temps excessif nécessaire à sa convergence.
Malik et Yu [152] ont aussi présenté un algorithme exacte de branch-and-bound pour ce problème avec une application de la relaxation lagrangienne et de l'algorithme du sous-gradient pour obtenir des bornes inférieures.

Comme pour tous les problèmes NP-complets, plusieurs heuristiques ont été dévéloppées pour résoudre le CSTP. Une des premières et des plus célèbres heuristiques pour le CSTP est celle d'Esau et Williams [58]. Il s'agit d'une méthode greedy qui part d'un arbre vide et ajoute au fur et à mesure les arêtes selon leurs coûts modifiés croissants sans violer la contrainte de capacité jusqu'à obtenir un arbre réalisable. Où, le coût modifié d'une arête $\{i, j\}$ est égale à son coût initial diminué du minimum des longueurs des chemins joignant le noeud racine 0 aux deux noeuds i et j.
En 1976, une des premières métaheuristiques a été présentée par Karnaugh [123]. En 2003, une application de la méthode de GRASP (Greedy Randomized Adaptive Search Procedure) a été proposée par Ahuja et al. [1]. Le principe de la recherche tabou a été appliqué par Sharaiha et al. [183] et aussi par Amberg et.al [3] qui ont obtenu d'excellents résultats.

2.6 Le problème de l'arbre généralisé de poids minimal

Definition 10 *Considérons un graphe non orienté $G = (V, E)$ et une partition $V_1, V_2, ..., V_K$ de l'ensemble des sommets V. A chaque arête $e \in E$ est associé un coût positif c_e.*

Le probème de l'arbre généralisé de poids minimum (Generalized Minimum Spanning Tree problem GMST) consiste à chercher un sous-graphe connexe de poids minimum qui doit inclure au moins un sommet de chaque sous-ensemble.

Si $|V_k| = 1, \forall k = 1, ..., K$, alors le GMST se réduit au problème classique d'arbre couvrant de poids minimal.
Dans ce qui suit, nous présentons quelques exemples d'application du problème de l'arbre généralisé de poids minimal.

Exemples d'applications du GMST

Une première application du GMST est la modélisation du problème d'interconnexion de réseaux. Etant donnés plusieurs quartiers d'une ville (les sous-ensembles), au sein de chaque quartier, il y a déjà un réseau local de lignes basse tension. Une étude peut révéler des possibilités de connexion des quartiers par des lignes moyenne tension. On cherche à interconnecter les diférents réseaux locaux de la manière la moins coûteuse possible. En terme de théorie de graphe, ceci revient à construire un arbre qui inclut au moins un sommet de chaque sous-ensemble.
Ce type de modélisation peut être appliqué à des réseaux de voie ferrée, d'eau potable ou encore à des centres de distribution.

Une deuxième application du GMST est la conception des réseaux d'irrigation dans un environnement aride [50]. Etant donné un ensemble de K parcelles partageant une même source d'eau. Chaque parcelle k ($k = 1, ..., K$) a une forme de polygone dont les sommets représentent le sous-ensemble de noeuds V_k. L'ensemble V_0 correspond au noeud modélisant la source d'eau. Le problème consiste alors à concevoir un réseau d'irrigation de longueur totale minimale qui connecte au moins un sommet de chaque parcelle. Ce qui revient au GMST qui garantie à chaque parcelle au moins une source d'irrigation.

Complexité du GMST

En 1995, Myung et al. [161] ont introduit une variante du $GMST$ appelée le $EGMST$. Il s'agit du cas où on n'impose qu'exactement un seul noeud de chaque sous-ensemble soit connecté à l'arbre recherché. Ils ont aussi démontré qu'il s'agit d'un problème NP-difficile [161]. Plus tard, Pop [173] a montré qu'il s'agit d'un problème NP-difficile même s'il est définit sur un graphe sous forme d'arbre. En 2000, Dror et al. [50] ont présenté le GMST et ont prouvé qu'il s'agit aussi d'un problème NP-difficile [49].
Un autre résultat important sur la complexité du GMST a été démontré par Myung et al. [161]. Il fait l'objet du théorème suivant:

Theorem 11 *Soit H une heuristique du GMST en un temps polynômial. Supposons que $P \neq NP$. Il n'existe aucune constante positive finie L vérifiant:*

$$\frac{Z_H(I)}{Z^*(I)} \leq L$$

Pour toute instance I avec $Z^*(I)$ and $Z_H(I)$ sont les valeurs respectives de la solution optimale et de la solution approchée de H.

Ce théorème exprime le fait qu'on ne peut pas trouver une solution heuristique avec garantie de performance en un temps polynômial, sauf si $P = NP$. Ainsi, en plus d'être NP-difficile, le fait de chercher une solution heuristique efficace pour le GMST est aussi NP-difficile.

Résolution du GMST

Depuis 1995, plusieurs formulations mathématiques en nombre entier ont été dévéloppées par Myung et al. [161] ainsi qu'une comparaison de leurs relaxations linéaires. En 2002, Feremans et al. [64] ont décrit huit formulations mathématiques pour le GMST et ils ont établi des relations de dominance entre les polytopes de leurs relaxations linéaires. Une nouvelle classe de contraintes valides a été prouvée comme facettes du polytope du GMST [63]. En 2006, d'autres formulations inspirées des problèmes de flot ont été décrites par Haouari et Chaouachi [100].

En 2001, Pop et al. [174] ont présenté un algorithme d'approximation de facteur $2p$ pour le cas du GMST où la taille des sous-ensembles est limité à p noeuds. Dans le papier [50], Dror et al. ont présenté un algorithme génétique pour le GMST qui donne des bornes supérieures avec un gap moyen de 6.53% par rapport à l'optimum. Plus récemment, en 2008 Hu et al. [110] ont proposé plusieurs variantes de la méthode de recherche du voisinage qu'ils ont appliquées sur des instances du GMST de taille atteignant 1280 noeuds avec d'excellents résultats.

En 2005, Haouari et al. [101] ont présenté un algorithme exact de branch-and-bound pour le GMST ainsi qu'une technique efficace de réduction du graphe. Feremans et al. [63] ont dévéloppé une heuristique du type recherche taboue et un algorithme excate de branch-and-cut pour le EGMST. Cette procédure a résolu de manière exacte des instances de taille atteignant 160 noeuds.

Dans le paragraphe suivant, nous présentons une généralisation récemment dévéloppée du problème de l'arbre généralisé.

2.6.1 Le problème de l'arbre généralisé avec collecte de prix

Comme son nom l'indique, le probème de l'arbre généralisé avec collecte de prix (Prize Collecting Generalized Minimum Spanning Tree PCGMST) est une *généralisation* du GMST. En effet, en plus des coûts associés aux arêtes, des poids sont associés aux noeuds. On cherche toujours un sous-graphe connexe de poids minimal connectant au moins un sommet de chaque sous-ensemble, sauf que le coût de connexion est la somme des coûts des arêtes et des poids des noeuds séléctionnés.

En 2004, Le PCGMST a été introduit par Stanojevic, Golden et Raghavan [201]. En 2008, les mêmes auteurs ont présenté plusieurs heuristiques basées sur le principe de la recherche locale et des algorithmes génétiques. Ils ont aussi dévéloppé un algorithme exacte de branch-and-cut qui a réussi à résoudre des instances de 200 noeuds [84].

En 2007, Pop [172] a présenté plusieurs formulations mathématiques pour le PCGMST auquelles il a établi une étude polyhédrale et il a comparé leurs relaxations linéaires. Il a aussi décrit une procédure exacte résolvant à l'optimalité des instances de 240 noeuds.

2.7 Le problème de Steiner dans les graphes

Le problème de Steiner dans les graphes (Steiner Tree Problem STP) est parmi les problèmes les plus étudiés de l'optimisation combinatoire. En effet, plusieurs centaines de papiers ont été consacrés à l'étude de ce problème. Autre que ses intérêts théoriques, ce problème trouve des applications dans divers domaines comme les réseaux de communication, la conception d'autoroutes, de pipelines, etc.

Nous commençons par présenter un aperçu historique du STP, suivi du problème de l'arbre de steiner. Bien que le problème d'arbre de Steiner soit de classe NP-complet, Il existe plusieus heuristiques qui ont été développées pour approcher la solution exacte dans un temps polynômial. Nous allons présenter en détails les deux heuristiques les plus connues de la littérature. Enfin, nous allons aussi présenter les cinq variantes les plus connues de la littérature.

2.7.1 Le problème de Steiner

Au cours du XVII siècle, le mathématicien français Pierre de Fermat chercha comment on pouvait trouver un point P dans un triangle ABC de manière à ce que la longueur totale des arêtes concourantes au point P soit la plus petite possible. Cette question fut résolue indépendamment par Fermat lui même et Torricelli et Cavalieri [85] qui établirent qu'un tel minimum est atteint soit par un sommet, soit par un point interne au triangle pour lequel les demi-droites $[PA)$, $[PB)$ et $[PC)$ forment des angles de 120 degrés.

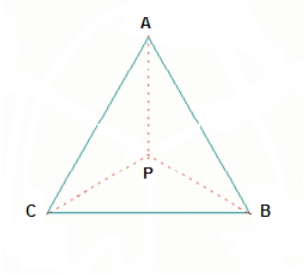

Figure 2.2: Exemple de problème de steiner sur un triangle équilatéral

Au XIX^{ième} siècle, un mathématicien de l'université de Berlin *Jakob Steiner* (1796-1863) généralisa ce problème à un ensemble de n points. Le problème revient à chercher à connecter n points du plan par un réseau connexe de longueur totale minimale. Ces points sont appelés les *points terminaux*. On montre facilement qu'un tel réseau est constitué uniquement de segments. D'autres points que les n sommets peuvent apparaître dans un réseau optimal: ils sont appelés les *points de Steiner*. C'est à partir de 1930 que le problème d'arbre de steiner dans les graphes a été considéré. Le nom de Steiner a été popularisé par Courant et Robbins dans la première édition de leur ouvrage datant de 1941 [41].

Nous citons quelques caractéristiques mathématiques de la solution du problème de Steiner [112]:

- Le degré d'un point terminal est au maximum égal à 3,
- Le degré d'un point de Steiner est égal à 3,

- L'angle entre deux arêtes incidentes au même point est au minimum de 120 degrés,
- Le nombre des points de Steiner est au maximum égal à $n-2$, où n est le nombre des points terminaux.

2.7.2 Le problème de l'arbre de Steiner

Definition 12 *Soit $G=(V,E)$ un graphe connexe non orienté. A chaque arête $e \in E$ est associé un coût positif c_e. L'ensemble des noeuds $V = T \cup S$ avec $T \cap S = \emptyset$. T est appelé l'ensemble des noeuds* terminaux *et S est l'ensemble des noeuds de* Steiner. *Cherchons l'arbre de poids minimal qui couvre* tous les sommets *de T et éventuellement des sommets de S.*

Ce problème est connu dans la littérature comme le problème de Steiner dans les graphes ou encore le problème de l'arbre de Steiner (Steiner Tree Problem STP).

Deux cas particuliers du problème de l'arbre de Steiner sont résolus par des algorithmes polynomiaux:

- Si $|T| = 2$, le problème de Steiner se réduit au problème du plus court chemin,
- Si $T = V$, le problème de Steiner se réduit au problème de l'arbre couvrant de poids minimal.

Un exemple illustratif du problème de Steiner est présenté par la figure 1.2 suivante:

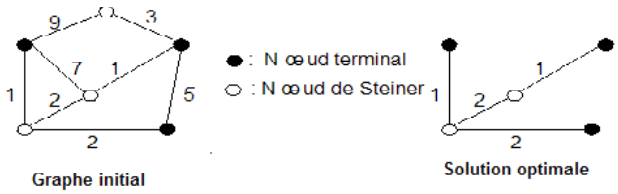

Figure 2.3: Exemple de problème de l'arbre de Steiner

Son NP-complétude a été établie en 1972 par Karp [124], mais ce résultat de complexité n'a pas empêché la recherche de méthodes de résolution efficaces en pratique.
La littérature sur ce problème est assez volumineuse. Nous renvoyons le lecteur à l'ouvrage de Hwang, Richards, et Winter [112] pour la fascinante histoire de ce problème. Pour plus de détails, nous citons les papiers de Hwang et Richards [111] et de Du et al. [51].

Exemples d'application du STP

Les applications du problème de Steiner sont nombreuses et diversifiées. Citons à titre illustratif les trois applications suivantes:

Optimisation des réseaux de télécommunication

On cherche à connecter des clients dans un réseau de télécommunication en utilisant un arbre de coût minimum. Si tous les noeuds du réseau sont des clients, le problème sera réduit au problème de l'arbre couvrant de poids minimal. Si on a besoin de connecter uniquement quelques utilisateurs "clés" avec des lignes ayant une haute fiabilité en utilisant éventuellement d'autres utilisateurs intermédiaires qui seront connectés par des lignes moins fiables, le problème est ainsi modélisé par le problème de Steiner.

Routage dans les circuits VLSI en électronique

Les concepteurs des circuits intégrés cherchent à connecter un ensemble de modules sur une surface de circuit intégré en minimisant la longueur totale du fil métallique utilisé. Il est à noter que les liens entre les fils utilisent des routes alignées suivant la direction de la surface Nord/Sud ou Est/Ouest. Par conséquent, la distance entre toute paire de modules est rectiligne.
Si on représente chaque position de module par un point, ce problème est un problème de Steiner où l'ensemble des points représentant les positions des modules est l'ensemble des noeuds terminaux T et la norme appliquée est $|.| \equiv L_1$.

Reconstruction de l'histoire évolutive en phylogénie

La phylogénie est l'étude de la formation et de l'évolution des organismes vivants en vue d'établir leur parenté. Un problème central de la phylogénie consiste à reconstruire l'arbre d'évolution d'un ensemble d'espèces biologiques. Le but est de retrouver l'arbre de vie : Si trois espèces A, B et C ont un ancêtre commun, dans quel ordre les différentiations de ces espèces

se sont-elles effectuées ?
L'explosion récente de la quantité de séquences d'ADN découvertes chez un grand nombre d'espèces offre la possibilité d'appréhender des questions complexes de la biologie allant à établir les liens de parenté entre les espèces. En effet, chaque espèce est représenté par un segment de son code ADN. Chaque segment d'ADN est identifié par une sequence de m lettres d'un sous-ensemble fini de lettres de l'alphabet qu'on note A (i.e., un vecteur de $V = A^m$). Ainsi, le problème est modélisé par un graphe complet $G = (V, E)$. A chaque paire de sequences (i, j), on associe "une distance" ou un coût positif c_{ij}. Le problème revient à déterminer un arbre de Steiner de coût minimal dans le graphe G, où l'ensemble des noeuds terminaux correspondent à l'ensemble d'espèces qu'on cherche à reconstruire leur histoire évolutive.

Comme il s'agit d'un problème NP-complet, il n'existe pas d'algorithme donnant une construction exacte d'arbre minimum de Steiner avec un temps polynômial. Par contre, plusieurs heuristiques permettant d'approcher la solution optimale ont été proposées. Un grand nombre de ces heuristiques reprend soit les idées de base utilisées pour la construction d'arbres couvrant de poids minimum, soit le principe de construction des plus courts chemins entre les noeuds pour construire un arbre couvrant partiel relativement avantageux.

Heuristiques basiques du STP

Comme le STP est NP-complet, plusieurs algorithmes heuristiques ayant une complexité polynômiale ont été proposés. Nous détaillons les deux heuristiques qui ont reçu le plus d'intérêt dans la littérature à savoir l'algorithme de Takahashi et Matsuyama et l'heuristique SDISTG.

Heuristique de Takahashi et Matsuyama (1981)

Une des heuristiques les plus simples du STP est l'algorithme de Takahashi et Matsuyama [185] qui reprend l'idée de Prim pour construire un sous-arbre couvrant.

L'algorithme de base proposé par Prim construit un arbre couvrant global minimum. En partant de l'état initial où un seul noeud arbitraire du graphe compose l'arbre couvrant, cet algorithme ajoute à l'arbre le noeud le plus proche jusqu'au dernier noeud du graphe. Takahashi et Matsuyama [185] ont adapté l'idée de Prim pour la construction d'un arbre couvrant partiel car seuls les noeuds terminaux doivent être couverts par la solution.

Le principe de l'algorithme de Takahashi et Matsuyama est le suivant:

Initialisation : Initialiser l'arbre en choisissant arbitrairement un noeud terminal.

Jusqu'au dernier noeud terminal

Calculer les distances des noeuds terminaux non encore choisis avec l'arbre ainsi obtenu.

Ajouter à l'arbre le noeud terminal le plus proche, en utilisant le plus court chemin entre le noeud et l'arbre.

L'algorithme admet une complexité de $O(pn^2)$ car les plus courts chemins peuvent être calculés par l'algorithme de Dijkstra [46] en $O(n^2)$, où n est le nombre total des noeuds et p est le nombre des noeuds terminaux.
L'arbre construit par cette heuristqiue est au plus *deux* fois plus long que la solution optimale [207]. Souvent, la solution approchée est relativement proche de l'optimum. Cependant, dans certains cas, le résultat fourni par toute heuristique utilisant les plus courts chemins reste éloigné de la solution optimale [160].

Heuristique Graphe avec la plus courte distance: SDISTG
Cette heuristique a été introduite par Kou et al. [133] en 1981. Cette heuristique appliquent les chemins les plus courts pour relier les noeuds terminaux.

Une description de l'heuristique SDISTG est la suivante:
Etape 1: Construire un graphe complet avec l'ensemble des noeuds terminaux et associer à chaque arête un poids correspondant au plus court chemin entre les paires des noeuds terminaux.
Etape 2: Construire un arbre couvrant de poids minimal sur ce nouveau graphe.
Etape 3: Remplacer chaque arête de l'arbre résultat de l'Etape 2 par le plus court chemin correspondant.
Etape 4: Soit $T = (V_T, E_T)$ le résultat du graphe de l'Etape 3.
4.1- Construire un arbre couvrant de poids minimal $T = (V'_T, E'_T)$ de T.
4.2- Tant qu'il existe une feuille de T' qui est un noeud de Steiner, alors enlever cette feuille ainsi que ses arêtes incidentes.
Etape 5: Supprimer les arêtes et les noeuds de Steiner de manière à ce qu'il n'existe aucun cycle et que toutes les feuilles sont des noeuds terminaux.

Depuis 1981, cette heuristique a suscité une grande attention dans la littérature ainsi que son implémentation. Cette heuristique a été testée sur des instances de taille atteignant 100 noeuds et 500 arêtes, avec un gap moyen par rapport à la solution exacte de 3,9% [133]. Cependant, même si le temps de calcul de ce genre de construction est toujours faible, la longueur des arbres résultants n'est pas toujours acceptable.

D'autres variantes d'heuristiques basées sur l'insertion des noeuds (ARINS, CHINS, CHINS-A) ont été dévéloppées [185]. Ces heuristiques ont suscité le plus grand intérêt théorique dans la littérature ainsi que la comparaison empirique de leurs performances [202].

Autres Heuristiques et approches de résolution du STP

La majorité des papiers concernant le STP proposent des méthodes approchées. De nombreuses métaheuristiques ont été appliquées pour apporter des solutions approchées à STP citons à titre d'exemple l'application de la méthode GRASP par Martins et al. [155] et aussi les algorithmes génétiques [59].
Le principe de la recherche tabou a été appliqué par Ribeiro et De Souza [179] en se basant sur une recherche locale avec le principe d'insertion et d'élimination des noeuds de Steiner. Cette approche donne des gaps meilleurs et au moins comparable aux autres heuristiques du STP, avec des temps de calcul nettement inférieurs.
Plusieurs algorithmes d'approximations ont été dévéloppés pour le STP [27]. A notre connaissance, le meilleur algorithme d'approximation pour le STP est celui de Robins et Zelikovsky [180] datant de 2005 et dont le facteur d'approximation est de $1 + \frac{\ln 3}{2} \approx 1.55$.

Parmis les méthodes exactes dévéloppées pour résoudre le SPT, nous citons un algorithme d'énumération introduit par Hakimi [95] et amélioré par Lawer [137]. En 1971, le principe de la programmation dynamique a été appliqué par Dreyfus et Wagner [48]. Il a été repris en 2002 par Cislik et al. [33]. Plusieurs algorithmes de type branch-and-bound et branch-and-cut ont été publiés [30], [130] et [146]. Parallèlement, des bornes inférieures ont été développées pour le STP. Nous citons à titre d'exemples le travail de Mamadi et Plesnik [153] et l'application de la relaxation lagrangienne par Beasely [16]. Polzin et Daneshmand [171] ont présenté une comparaison, en

terme de relaxation linéaire, des différentes formulations mathématiques du STP.

Le problème de Steiner présente plusieurs variantes selon la nature de l'application réelle qu'il modélise. En effet, dans la littérature, il existe une vingtaine de variantes de ce problème. Dans la suite, nous présentons les cinq variantes les plus étudiées du STP, à savoir, le problème de Steiner rectiligne, le problème de Steiner euclidien, le problème de Steiner généralisé, le problème de Steiner arbre-étoile et le problème de Steiner avec contrainte de délais.

2.7.3 Le problème de Steiner euclidien

En 1968, une version du problème de Steiner sur un plan bidimensionnel a été introduite par Cockayne et Melzak [34]. Elle a été examinée plus tard par Weng [206] en 1985. En 1990, Reich et Widmayer [177] était les premiers à considérer ce problème sur les graphes. En effet, Si T_i et T_j sont deux points du plan de coordonnées respectives (x_i, y_i) et (x_j, y_j), alors le coût de l'arête connectant T_i et T_j est la distance euclidienne $c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. Il s'agit d'une variante du problème de Steiner qu'on appelle le problème de Steiner euclidien (Euclidean Steiner Tree Problem ESTP).

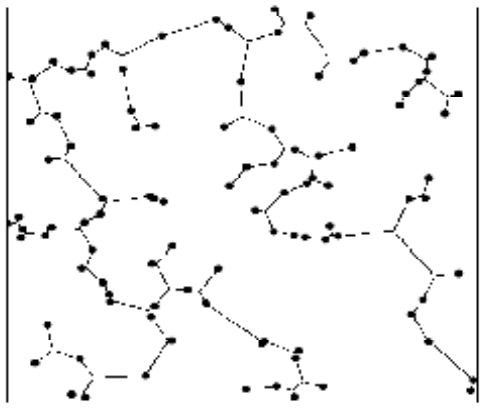

Figure 2.4: Exemple de solution optimale du problème de Steiner euclidien avec 100 noeuds terminaux [205]

En 2000, Maculan et al. [149] ont présenté une formulation mathématique avec des variables mixtes à laquelle ils ont appliqué la relaxation lagrangienne pour obtenir une borne inférieure au ESTP.

Plusieurs métaheuristqiues ont été proposées pour le ESTP telles que les algorithmes génétiques [12] et le principe de la recherche locale [47]. En 2006, Yang [211] a dévéloppé une heuristique hybride combinant les algorithmes génétiques et la recherche locale assez efficace avec un gap de 0.3% par rapport à l'optimum.

2.7.4 Le problème de Steiner rectiligne

Soient T_i et T_j deux points du plan de coordonnées respectives (x_i, y_i) et (x_j, y_j). Le coût de l'arête connectant T_i et T_j est donnée par $c_{ij} = |x_i - x_j| + |y_i - y_j|$.
Notons T l'ensemble de sommets appelés les points terminaux. Le problème de Steiner rectiligne (Rectilinear Steiner Tree Problem RSTP) consiste à connecter les points terminaux en minimisant la somme des arêtes séléctionnés. Comme pour le problème de Steiner, il suffit de chercher un arbre de Steiner de poids minimal en utilisant un ensemble de noeuds facultatifs S qui sont les noeuds de Steiner.

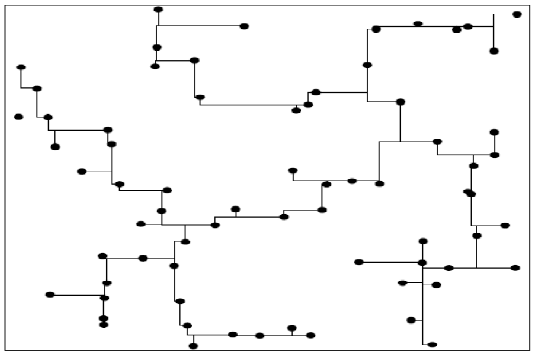

Figure 2.5: Exemple de solution optimale du problème de Steiner rectiligne avce 70 noeuds terminaux [205]

Parmi les applications de ce problème, on peut citer le routage sur les plaques des circuits électriques (VLSI) et l'aménagement des lignes électriques tout le long des avenues d'une ville.

Depuis 1966, le problème de Steiner rectiligne a été formulé par Hanan [97] qui a aussi prouvé qu'on peut le transformer en un problème d'arbre de Steiner classique. En effet, si une grille est dessinée avec les lignes horizontales et verticales traversant tous les points terminaux, alors l'ensemble des points de Steiner serait les noeuds (autre que les points terminaux) de cette grille qu'on appelle la grille de Hanan [97].

Même si tout algorithme résolvant le STP permet la résolution du problème de Steiner rectiligne, un algorithme exploitant les particularités du problème de steiner rectiligne a été dévéloppé par Ganley et Cohoon [73]. Un autre algorithme exact de type branch-and-cut a été dévéolppé par Warme [205]. Ce dernier a obtenu en des temps de calcul raisonnable l'optimum pour des instances avec 500 points terminaux.

Comme il s'agit d'un problème NP-Complet [74], plusieurs heuristiques ont été dévloppées telles que celle de Beasley [15] et De Souza et Ribeiro [44]. En 2005, Yang [212] a présenté une heuristique inspirée de l'algorithme de Prim qui a donné d'excellents résultats dépassant les autres heuristiques déjà dévéloppées pour le RSTP.

En 2003, Zachariasen et Rohe [214] ont présenté un algorithme exacte de résolution d'une *généralisation* du problème de Steiner rectiligne. Il s'agit du cas particulier du RSTP où les sommets du graphe sont répartis en sous-ensembles contenant chacun au moins un noeud terminal. Le RSTP généralisé consiste à déterminer une connexion avec moindre coût connectant au moins un sommet de chaque sous-ensemble. L'algorithme construit essentiellement un sous-graphe selon la grille de Hanan [97] sur laquelle on résoud le problème de l'arbre de Steiner classique. Des instances modélisant des problèmes réels composés de 100 sous-ensembles ont été ainsi résolus de manière exacte.

2.7.5 Le problème de Steiner généralisé

Definition 13 *Soient $G = (V, E)$ un graphe connexe non orienté et $T \subseteq V$ telque T soit partitionné en K sous-ensembles $V_1, ..., V_K$. Le problème de Steiner généralisé consiste à chercher un arbre de G ayant un coût minimal et qui contient au moins un noeud de chaque sous-ensemble V_k, $k = 1, ..., K$. Des noeuds de Steiner du sous-nsemble $V \setminus T$ peuvent éventuellement être couvert par l'arbre de poids minimal recherché.*

Dans la littérature, ce problème est aussi connu sous le nom du CSTP (Class Steiner Tree Problem).

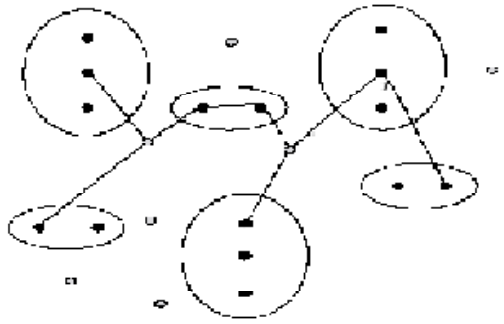

Figure 2.6: Exemple de solution réalisable du problème de Steiner généralisé

En 1990, Le CSTP a été introduit par Reich et Widmayer [177]. Ils ont aussi indiqué qu' il s'agit de l'application directe du problème de routage dans les circuits éléctroniques VLSI avec des modules à ports multiples, contrairement au modèle traditionnel où chaque module admet un port unique. En effet, chaque module est une collection de différents ports. Un module peut être conecté au réseau par un de ses ports. Le problème consiste à déterminer le réseau de longueur totale minimale qui connecte au moins un port de chaque module.

Le CSTP est un problème NP-difficile car il s'agit d'une généralisation du problème de steiner (STP) qui est NP-difficile. Ihler et al. [114] ont montré que le CSTP est NP-difficile même si les noeuds et les arêtes du graphe forment une grille unitaire dans le plan et que les sous-ensembles admettent au plus 3 noeuds.
Seul le cas particulier de CSTP, où les sous-ensembles V_k, $k = 1, .., K$, sont des intervalles de points sur deux droites parallèles, est de complexité polynômiale [113]. En 1999, Ihler et al. [114] ont montré que grace à une transformation du graphe, le CSTP peut être reformulé en un problème équivalent au problème de l'arbre de Steiner classique (STP). Haouari et al. [101] ont montré que le CSTP peut être reformulé comme un problème d'arbre généralisé de poids minimal.

En 2000, Salazar [182] a introduit une nouvelle formulation en nombre entier pour CSTP et il a présenté une étude polyédrale. En utilisant une technique de "lifting", il a permis l'adaptation des facettes du problème de l'arbre couvrant de poids minimal pour le CSTP. Ainsi, il a établi de nouvelles classes d'inégalités qui définissent des facettes pour le polyèdre du CSTP. En 2006, ces résultats ont été repris et améliorés par Ferreira et Filho [65] qui ont aussi dévéloppé de nouvelles formulations plus développées pour le CSTP.

Pour ce qui concerne le développement de borne inférieure pour le CSTP, nous citons l'application de la relaxation lagrangienne proposée par Yang et Gillard en 2000 [213].

Plusieurs algorithmes d'approximation ont été dévéloppés dans la littérature, nous citons les papiers de Helvig et al. [107] et plus récement Duin et al. [53] qui ont présenté une heuristique efficace pour le CSTP basée sur l'idée de Salsar [182] de transformer le CSTP en un STP sur un graphe élargie.
En 2006, Chekuria et al. [29] ont présenté un algorithme glouton récursif pour le CSTP qui est largement inspiré de l'algorithme d'approximation dévéloppé par Charikar et al. [27] pour le problème de l'arbre de steiner.

2.7.6 Le problème de Steiner arbre-étoile

Le problème de Steiner arbre-étoile ou encore le problème de Steiner Tree-Star (STS) a été introduit par Lee et al. [138] depuis 1994.

Considérons un graphe complet $G = (V, E)$. Un coût $c_{i,j}$ est associé à chaque arrête $(i, j) \in E$ et un poids b_j est associé à chaque noeud $j \in V$. L'ensemble des noeuds V est partitionnés en deux sous-ensembles T et S. Supposons qu'il existe m noeuds dans le plan appelés *les noeuds terminaux* et indexés par $i \in T = \{1, ..., m\}$. Ces noeuds peuvent être connectés à n autres noeuds, appelés les noeuds de Steiner indexés par $j \in S = \{1, ..., n\}$. Chaque noeud de Steiner $j \in S$ connecté à au moins un noeud terminal ou à un autre noeud de Steiner, est appelé *noeud de Steiner actif*. Chaque noeud terminal $i \in T$ doit être connecté à exactement un noeud de steiner actif $j \in S$, avec un coût de connexion c_{ij}. En plus, deux noeuds distincts de Steiner j et $k \in S$ peuvent être directement connectés avec un coût c_{jk}, alors que deux noeuds terminaux ne peuvent pas être connectés directement.
Nous rappelons qu'une étoile ou Star est un sous-garphe de G constitué d'un noeud de Steiner (au centre de l'étoile) connecté à des noeuds cibles (situés

autour).
Le problème Steiner arbre-étoile consiste à déterminer l'arbre de coût minimal qui couvre tous les noeuds terminaux en passant par des noeuds de Steiner actifs.

Ce problème est assez intéréssant sur le plan théorique car il s'agit d'un problème NP-difficile. Sur le plan pratique, ce problème s'applique au niveau de la conception des réseaux à données digitales. Le STS peut être considéré comme un problème de Steiner avec d'une part des poids associés aux nombres et d'autre part des contraintes de degrés sur les noeuds terminaux.

Lee et al. [139] ont présenté une formulation mathématique avec des variables mixtes pour le STS. Ils ont dévéloppé une heuristique ainsi qu'une procédure exacte de type branch-and-cut pour résoudre des problèmes de taille allant à 200 noeuds.

Une heuristique de type recherche tabou a été développée par Xu et al. [210] qui ont trouvé que la performance de leur approche est meilleure que celle d'une simple heuristique basée sur la recherche locale.
En 2002, Khuller et Zhu [127] ont présenté deux algorithmes polynomiaux d'approximation de facteurs respectifs 5.16 and 5. Ces algorithmes ont été appliqués sur une version généralisée du STS qui n'est d'autre que le NWST (Node-Weighted Steiner Tree) que nous présentons dans le paragraphe 2.8.1.

2.7.7 Le problème de Steiner avec contrainte de délai

Definition 14 *Soit un graphe non orienté $G = (V, E)$ tel que $V = S \cup T$ avec $S \cap T = \emptyset$. L'ensemble des noeuds terminaux T contient un noeud source r. Un cout c_{ij} et un délai d_{ij} sont associés à chaque arête $(i, j) \in E$. Le problème consiste à déterminer l'arbre de poids minimal qui recouvre* tous les sommets *de T et éventuellement des sommets de S et telque la somme des délais sur chaque chemin P_j reliant la source r au noeud terminal $j \in T\backslash\{r\}$, ne dépasse pas un retard ,ou délai, maximal fixé Δ.*

Ce problème est connu dans la littérature sous le nom du problème de Steiner avec contrainte de délai (Steiner Tree Problem with Delays STPD).

Le problème de Steiner classique (STP) a été généralisé en le STPD sous l'exigence du concept de la qualité de service (Quality of Service, Qos) dans la conception des réseaux de communication. Citons l'exemple des réseaux

multicast, pour les quels on n'a pas uniquement le souci de minimiser le coût de connexion mais aussi il faut que le délai de transfert d'un message envoyé par la source vers les membres du réseau ne dépasse pas un temps maximal de transmission à travers le réseau [132].

Il est assez facile de vérifier que le problème de l'arbre de Steiner (STP) est un cas particulier du STPD. En effet, il suffit de considérer le noeud source r comme tout noeud terminal et de prendre le retard maximal $\Delta = +\infty$. Par conséquent, on montre que le STPD est un problème NP-difficile.

Une formulation mathématique en nombres entiers du STPD a été présentée ainsi qu'une méthode exacte dans le papier de Noronha et Tobagi [164] qui ont résolu à l'optimum des instances de petites tailles (maximum de 12 noeuds). En 2007, Leggieri et.al [141] ont présenté plusieurs formulations compactes, ainsi qu'une heuristique et une technique de réduction de graphe qui ont permis de résoudre exactement des instances de taille atteignant 1000 noeuds sur des graphes de faible densité.

En 2006, Tseng et al. [186] ont dévéloppé un algorithme génétique pour le STPD. Kompella et al. [132] ont présenté une heuristique de type Greedy dont l'idée est de construire l'arbre couvrant sur un graphe élargi et de chercher les plus courts chemins avec contrainte de délai entre la source et les autres noeuds terminaux.

2.8 Le Problème de Steiner avec Collecte de Prix

Il s'agit d'une généralisation du problème de Steiner classique auquel on associe des poids aux sommets du graphe et on impose des contraintes suplémentaires. Une revue de la littérature assez détaillée sur les différentes variantes du PCSTP est disponible dans le récent papier de Costa et al. [40]. Nous nous intéressons aux quatre variantes les plus traitées dans la littérature.

2.8.1 La variante NWSTP

Depuis 1987, une variante du PCSTP, connue dans la littérature par le "Node-Weighted Steiner Tree Problem" qu'on note NWSTP, a déjà été

introduite par Segev [184].

Definition 15 *Soit un graphe $G = (V, E)$ tel que $V = S \cup T$ avec $S \cap T = \emptyset$. A chaque sommet $i \in V$, on associe un poids positif w_i et à chaque arête $e \in E$, on associe un coût positif c_e.*

Le NWSTP consiste à trouver un sous-arbre $\mathrm{T}' = (V', E')$ de G qui couvre tous les noeuds de T ($T \subset V'$) et qui minimise le coût des arêtes de T' diminué de la somme des poids des noeuds n'appartenant pas à T'.

La fonction objectif du NWSTP est de la forme suivante:

$$NS(T') = Min \sum_{e \in E'} c_e - \sum_{j \notin V'} w_j$$

Il évident que si les poids des noeuds sont nuls alors le NWSTP revient au problème d'arbre de Steiner classique.

Segev [184] a dévéloppé deux formulations du NWSTP basées sur les flots aux quelles il a appliqué la relaxation lagrangienne afin d'obtenir des bornes inférieures pour ce problème. A la suite, une procedure de branch-and-bound a été appliquée et testée sur des instances de taille atteignant 40 noeuds.

En 1998, Engevall et al. [55] ont introduit une nouvelle formulation du NSP comprenant une famille de contraintes d'élimination de sous-tours dont le nombre est exponentiel. Ils ont dévéloppé une heuristique basée sur la relaxation lagrangienne de ces contraintes et la construction de solutions réalisables en ne prenant en considération que les contraintes violées. Ils ont testé cette procédure sur des instances avec des graphes complets de taille atteignant 100 noeuds. En 2006, Cordone et Trubian [37] ont dévéloppé un algorithme exacte de branch-and-bound pour le NSP basé sur cette même formulation d'Engevall et al. [55]. Ils ont résolu à l'optimum des instances avec 1000 noeuds.
En 1995, Klein et Ravi [128] ont dévéloppé un algorithme d'approximation glouton pour le NWSTP. Cet algorithme a été généralisé et amélioré par Guha et Khuller [92] en 1998. Même si ces algorithmes sont polynomiaux, ils ne sont pas pratiques parce qu'ils demandent un temps de calcul assez long.

En 2006, une variante assez intéréssante sur le plan pratique du NWSTP a été abordé par Angelopoulos [4]. Il s'agit du cas où on impose une contrainte supplémentaire sur les poids des sommets couverts par la solution

optimale. En effet, le ratio du poids maximal par rapport au poids minimal ne doit pas dépasser une certaine constante. Cette variante s'applique dans le problème de routage où on impose une certaine homégénité en terme de coût des routeurs choisis dans le réseau. Angelopoulos [4] a dévéloppé et testé deux algorithmes d'approximation pour cette variante du NWSP.

2.8.2 La variante Goemans-Williamson

Definition 16 *Soit un graphe $G = (V, E)$. A chaque sommet $i \in V$, on associe un poids positif γ_i et à chaque arête $e \in V$, on associe un coût positif c_e. La variante Goemans-Williamson du PCSTP revient à déterminer un arbre $T = (V_T, E_T)$ de G qui minimise le coût des arêtes de T et la somme des poids des noeuds n'appartenant pas à T.*

La fonction objectif du problème est de la forme suivante:

$$GW(T) = Min \sum_{e \in E_T} c_e + \sum_{j \notin V_T} \gamma_j$$

En 1997, Goemans et Williamson [82] ont introduit cette version du PCSTP et ils ont aussi proposé un algorithme de 2-approximation avec une complexité de $O(n^2 \log n)$, où n est le nombre de sommets du graphe G. Cet algorithme a été repris et amélioré par plusieurs auteurs tels que Johnson et al. [116] qui ont proposé une nouvelle stratégie d'élimination d'arêtes et Cole et al. [35] qui ont développé une implémentation optimisée qui accélère sa convergence. Plus récemment en 2007, Melkonian [156] a développé une méthode de points intérieurs basée sur l'algorithme proposé par Goemans and Williamson [82] et qui a donné d'excellents résultats nettement meilleurs que ceux de l'algorithme de 1997.

En 2001, Canuto et al. [23] ont proposé un algorithme de recherche locale avec perturbation et avec des points de départ multiples. Les perturbations ont été accomplies en changeant les paramètres du graphe et en annulant les poids des noeuds potentiels. Les solutions réalisables correspondent à ceux obtenus par l'algorithme de Goemans-Williamson [82]. A la suite, on applique une procédure de recherche locale dont le principe est de partir d'une solution qu'on tente d'améliorer itérativement. Lors d'une itération, la solution courante est légèrement modifiée afin d'obtenir une solution voisine. Cet algorithme approche semble déterminer la solution optimale pour la majorité des instances testés.

En 2004, Lucena et Resende [147] ont présenté pour cette variante une formulation mathématique en nombre entier avec une preuve empirique que résoudre la relaxation linéaire de cette formulation donne d'excéllents bornes inférieures qui ont été utlisés lors d'algrithmes exactes de branch-and-bound et de branch-and-cut.

2.8.3 La variante NWMP

Une variante du PCSTP assez similaire à la variante Goemans-Williamson est connue sous le nom du problème de maximisation du profit net (Net Worth Maximization Problem NWMP)[116]. Comme son nom l'indique, cette variante consiste à déterminer un sous-arbre qui maximise le profit net collecté.

Definition 17 *Soit un graphe $G = (V, E)$. A chaque sommet $j \in V$, on associe un poids positif γ_j et à chaque arête $e \in V$, on associe un coût positif c_e. Le NWMP consiste à trouver un arbre $T = (V_T, E_T)$ de G qui maximise le profit total collecté diminué du coût des arêtes connectant T.*

La fonction objectif du NWMP est de la forme suivante:

$$NW(T) = \max \sum_{j \in V_T} \gamma_j - \sum_{e \in E_T} c_e$$

Il s'agit d'un problème de maximisation équivalent à la variante Goemans-Williamson du PCSTP. En effet, pour tout sous-arbre T de G, on a $NW(T) + GW(T) = \sum_{j \in V_T} \gamma_j$. Sauf que contrairement à la variante Goemans-Williamson, il n'existe pas d'algorithmes d'approximation pour cette variante. En effet, Feigenbaum et al. [61] ont montré qu'il est NP-difficile de trouver un algorithme d'approximation avec un facteur constant pour le NWMP.

En 2006, Ljubic et al. [148] ont présenté un algorithme de branch-and-cut qui se base sur une formulation mathématique du problème avec des contraintes de connectivité. Ces inégalités sont générées efficacement grâce à un algorithme de flot maximal.

2.8.4 Le problème de Quota

Le problème de Quota est une variante du problème de Steiner avec collecte de prix introduite par Johnson et al. [116]. Il correspond au problème

modélisé par un graphe, où l'ensemble des noeuds représente des localités qui doivent être interconnectées par des liens indiquant les différentes connexions entre ces locaux, tout en sachant qu'à chaque noeud est associé un profit. Le problème consiste alors à déterminer un arbre de coût minimal de telle sorte que le total des profits des noeuds de cet arbre soit au minimum supérieur ou égal à un Quota prédéfini.

Comparée aux autres versions du PCSTP , cette version a reçu une attention assez modeste dans la littérature. Dans leur papier [116], Johnson et al. ont observé qu'il s'agit d'une généralisation du problème k-MST (parapgraphe 2.4) en associant des profits unitaires aux sommets du graphe et en fixant le quota à collecter à l'entier k. Partant de ce constat, ils s'inspirent de l'algorithme de Arya et Ramesh [6] pour développer un algorithme polynômial d'approximation avec un facteur de 2.5 pour cette variante du PCSTP.

En 2006, Haouari et Chaouachi [99] ont développé un algorithme génétique hybride. En effet, la relaxation lagrangienne a été appliquée avec l'algorithme du volume (voir paragraphe 4.4.2) à la place de l'algorithme du sous-gradient classique. Haouari et Chaouachi présentent aussi une modification de l'algorithme de Prim [175] pour obtenir des solutions réalisables au problème. Cette procédure a été testée sur des instances de taille atteignant 500 noeuds et 5000 arêtes et a donné un gap inférieur à 2% pour 76% des instances résolues.

2.9 Le problème de Steiner avec collecte de prix généralisé

En 1993, le PCSTP a été formulé pour la première fois par Bienstock et al. [17] qui ont proposé un algorithme d'approximation pour une version simplifiée de ce problème. Mais, la première référence à un problème d'optimisation combinatore *avec collecte de prix* est due au travail de Balas qui date de 1989 [7]. Balas a introduit le problème de voyageur de commerce avec collecte de prix (*Prize Collecting Traveling Salesman Problem* PCTSP). Dans le PCTSP, l'objectif est de déterminer un circuit, non nécéssairemnt hamiltonien, dans un graphe contenant un noeud racine, tel que la somme des couts des arêtes ainsi que des pénalités est minimisée et la somme des

profits collectés est au moins égale à un quota fixe. Pour une récente revue de littérature sur les différentes variantes du problème de voyageur de commerce avec profits, nous citons le papier de Feillet et al. [62].

Le problème de Steiner avec collecte de prix généralisé (GPCSTP) est un nouveau problème d'optimisation combinatoire dont les données sont les suivantes:

- *un graphe non orienté* $G = (V, E)$ *où l'ensemble de sommets est* $V = \{0, 1, ..., n\}$ *et* E *est l'ensemble des arêtes du graphe* G*. En plus,* 0 *est un noeud particulier qu'on notera le noeud racine.*
- *une partition* $(V_k)_{k=1...K}$ *tel que* $V^* = V \backslash \{0\} = V_1 \cup V_2 .... \cup V_K$ *avec* $V_i \cap V_j = \varnothing \ \forall i \neq j,\ i, j = 1...K$.
- *un ensemble de réels positifs* $(c_e)_{e \in E}$ *où* c_e *est le coût de l'arrête* e, $\forall e \in E$.
- *un vecteur de réels positifs* $(\gamma_j)_{j \in V^*}$ *où* γ_j *est la pénalité associée au sommet* j, $\forall j \in V^*$.
- *un vecteur de réels positifs* $(p_j)_{j \in V^*}$ *avec* p_j *le profit associé au sommet* j, $\forall j \in V^*$.
- *un vecteur de réels positifs* $(Q_k)_{k=1...K}$ *avec* Q_k *le quota associé au sous-ensemble* V_k, $\forall k = 1...K$.

On aborde une version généralisée du probème de Steiner avec collecte de prix (GPCSTP) qui consiste à déterminer un arbre $\tilde{G} = (\tilde{V}, \tilde{E})$ du graphe G vérifiant les trois proriétés suivantes:

(P1) Le noeud racine 0 appartient à $\tilde{V}$;

(P2) La somme des profits des noeuds de chaque sous-ensemble $\tilde{V} \cap V_k$, $k = 1, ..., K$, est supérieur ou égale au quota Q_k;

(P3) $\left(\sum_{e \in \tilde{E}} c_e + \sum_{j \in V \backslash \tilde{V}} \gamma_j \right)$ est minimum.

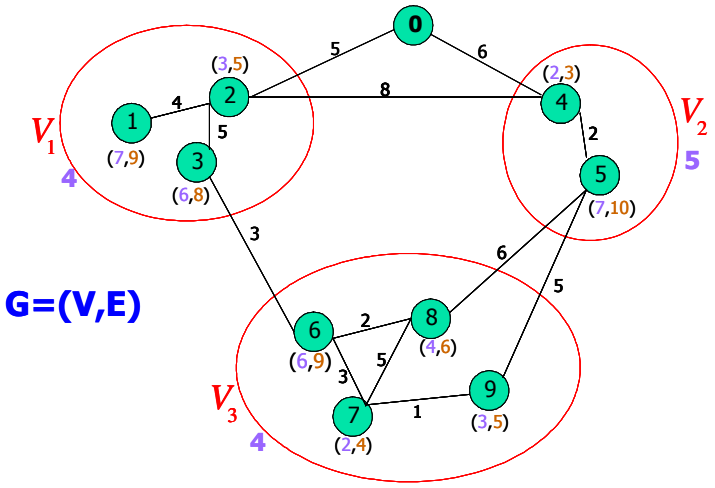

Figure 2.7: Exemple du problème de Steiner avec collecte de prix généralisé

Le but du problème de Steiner avec collecte de prix généralisé (GPCSTP) est de trouver un sous-ensemble de sommets S de V contenant le noeud 0 et tel que la somme du poids de l'arbre de poids minimal couvrant S dans le graphe G et des pénalités des noeuds n'appartenant pas à S, soit minimale. En plus, il faut que l'arbre proposé respecte la condition qu'au sein de chaque sous-ensemble V_k, la somme des profits des sommets connectés à l'arbre soit supérieure ou égale au quota Q_k.

2.9.1 Equivalence des variantes avec et sans racine

Comme pour la majorité des problèmes d'optimisation d'arbre, le GPCSTP admet une version avec racine et une autre sans racine. Tel qu'on l'a présenté dans le paragraphe précédent, le GPCSTP admet un noeud racine 0 qui doit appartenir à la solution optimale. Il s'agit de la version rooted qu'on note (P_r). Comme son nom l'indique, la version unrooted qu'on note (P) n'admet pas de noeud racine et considére V^* comme ensemble de sommets du graphe G.

Théorème 18 *Les problèmes* (P) *et* (P_r) *sont équivalents.*

Preuve. Sachant résoudre le problème (P_r) , on sait résoudre le problème (P). En effet, on peut procéder par deux manières différentes qui sont les suivantes:

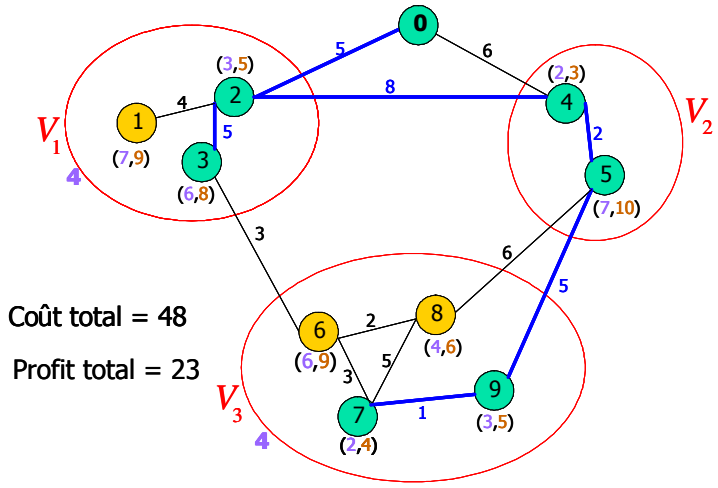

Figure 2.8: Exemple de solution réalisable du problème de Steiner avec collecte de prix généralisé

Première possibilité : On résoud n fois le problème (P_r) en fixant à chaque fois comme noeud racine un noeud i, $i = 1, ..., n$. Ensuite, on prend comme solution du problème (P) la meilleure solution parmis des n trouvés.

Deuxième possibilité : On ajoute un noeud fictif qu'on note 0 avec $p_0 = 1$ et $\gamma_0 = 0$. On ajoute un ensemble fictif $V_{K+1} = \{0\}$ avec $Q_{K+1} = 1$. On connecte le noeud 0 aux n noeuds réels avec des arêtes ayant un coût très élévé. Ainsi, si on résoud le problème (P_r), le noeud 0 sera connecté à un seul noeud de l'ensemble V^*. Il suffit d'éliminer le sommet 0 et l'unique arêtre qui le relie à l'arbre, pour retrouver la solution du problème (P).

Sachant résoudre le problème (P) , on peut résoudre le problème (P_r). En effet, ayant le noeud racine 0 , on crée un sous ensemble $V_{K+1} = \{0\}$ tel que $Q_{K+1} = 1$. Il suffit d'ajouter $p_0 = 1$ et $\gamma_0 = 0$. On retrouve ainsi les données du problème (P). ■

Dans ce présent travail, nous avons choisi de considérer le GPCSTP dans sa version avec racine parce qu'elle est la plus proche des applications réelles. Dans la partie suivante, nous présentons les intérêts pratiques et théoriques qui ont motivé notre choix.

2.9.2 Intérêts du problème

Comme le GPCSTP est un problème de conception de connexion de réseaux, il ne peut être que d'un intérêt pratique incontestable. En plus, nous montrons que ce problème généralise plusieurs problèmes bien connus d'optimisation de structures arborescentes.

Les intérêts pratiques du problème

Le PCSTP a plusieurs applications, mais celle qui nous a motivé lors de ce travail est son application au problème de conception des réseaux locaux (Local Area Network (LAN)). Prenons l'exemple d'un fournisseur d'accès et de services Internet qui cherche à installer un réseau en fibre optique pour fournir l'Internet à haut débit sur un domaine géographique donné. Partant d'une source fixe appellée racine, il doit interconnecter K gouvernorats, où chaque gouvernorat est composé de plusieurs villes. La compagnie ne peut installer les câbles entre deux villes que le long de routes bien définies dont les longueurs contribuent au coût de connexion de ces deux villes.

Dans la conception de ce réseau, le graphe correspondra au plan des routes de toute la région. Les noeuds correspondent aux villes et les arêtes correspondent aux segments de rues entre deux villes différentes. Le coût d'une arête est évidemment le coût d'installer un câble le long de cet arête.

En plus, les écomonistes de l'entreprise estiment pour chaque ville, un profit associé à sa consommation si elle est connectée au réseau et un manque à gagner si elle ne l'est pas. Il est évident que le manque à gagner associé à une ville est supérieur à son profit car l'image de marque du fournisseur sera dégéradée s'il refuse de connecter une ville à son réseau.

La compagnie n'est pas obligée de connecter toutes les villes au réseau mais elle a la possibilité de sélectionner les villes qui seront connectées. La compagnie cherche également à se garantir un profit minimum au sein de chaque gouvernorat.

En prenant K comme nombre des gouvernorats à connecter, on considère que les sommets du sous-ensemble V_k sont les villes du $k^{ième}$ gouvernorat et Q_k est le profit minimum à collecter au sein de ce gouvernorat. En plus, en notant γ_j et p_j respectivement la pénalité et le profit associé la $j^{ième}$ ville,

et c_{ij} le coût de connexion des villes i et j, on retrouve les notations du problème tel que nous avons décrit au paragraphe 3.2.

Les intérêts théoriques du problème

Outre ses intérêts pratiques, le problème que nous traitons est d'un intérêt particulier dans la théorie des graphes puisqu'il représente une généralisation de plusieurs problèmes classiques largement traitées dans la littérature. En effet, il est un cadre unificateur de plusieurs problèmes de classes NP-difficile.

Dans la suite, nous exposons les problèmes qui sont des cas particuliers du problème ($GPCSTP$) que nous traitons. Pour chaque problème, nous présentons les parmaètres du GPCSTP à fixer pour le retrouver.

Parmis ces problèmes, seul le problème de l'arbre couvrant de poids minimal est de classe P, tous les autres sont NP-difficiles.

Les problèmes	**Les paramètres du GPCSTP**
Le problème d'arbre couvrant de poids minimal	- $Q_k = \sum_{j \in V_k} p_j, \forall k = 1...K$, - $\gamma_j = +\infty,\ \forall j \in V^*$.
Le problème d'arbre de cardinalité k	- $K = 1$, - $V_1 = V^*$, - $Q_1 = k$, - $p_j = 1,\ \forall j \in V^*$, - $\gamma_j = 0,\ \forall j \in V^*$.
Le problème de l'arbre généralisé	- $Q_k = 1\ \forall k = 1, ..., K$, - $p_j = 1,\ \forall j \in V^*$, - $\gamma_j = 0,\ \forall j \in V^*$.
Le problème de l'arbre de Steiner	- $K = 2$, - $V_1 = T, V_2 = S$, - $Q_1 = \lvert T \rvert, Q_2 = 0$, - $p_j = 1\ \forall j \in V_1, p_j = 0\ \forall j \in V_2$, - $\gamma_j = 0\ \forall j \in V^*$.
Le problème de Steiner généralisé	- $\bigcup_{k=1,...,K} V_k = T, V_K = V \setminus T$, - $Q_k = 1\ \forall k = 1, ..., K-1, Q_K = 0$, - $p_j = 1\ \forall j \in T, p_j = 0\ \forall j \in V_K$, - $\gamma_j = 0\ \forall j \in V^*$.
Le problème du Quota	- $K = 1$, - $\gamma_j = 0, \forall j \in V^*$.
La variante NWMP	- $K = 1$, - $Q_1 = 0$, - $p_j = \gamma_j, \forall j \in V^*$.
La variante NWSTP	- $K = 1$, - $Q_1 = 0$, - $T = \{1\}$, - $\gamma_j = -w_j, \forall j \in V^*$.
La variante Goemans-Williamson	- $K = 1$, - $Q_1 = 0$.

2.10 Conclusion

Compte tenu de l'importance des applications en réseaux et en architecture des ordinateurs, la recherche de structures arborescentes optimales forme une famille de problèmes combinatoires qui ont été intensivement étudiés durant ces dernières décennies.

Ce chapitre a été consacré à la présentation de seize problèmes d'optimisation d'arbres les plus traités dans la littérature. Pour chaque problème, nous avons cité les principaux travaux et résultats qui ont été publiés. Vu son importance historique et ses multiples applications pratiques, le problème de Steiner ainsi que ces multiples généralisations ont été abordées de manière approfondie. Comme la majorité de ces problèmes sont NP-difficiles, la majorité d'algorithmes proposés dans la littérature sont plutôt des algorithmes d'approximation, alors que nous proposons des méthodes exactes de résolution du PCSTP.

Nous avons aussi introduit le problème de Steiner avec collecte de prix généralisé (Generalized Prize Collecting Steiner Tree Problem, GPCSTP) et nous avons exposés ses applications réelles ainsi que sa généralisation d'une dizaine de problèmes de structures arborescentes. En effet, le GPCSTP n'est d'autre qu'un cadre unificateur des problèmes définis sur les structures arborescentes les plus traités dans la littérature.

Dans le chapitre suivant, nous présenterons une technique de détermination de borne inférieure pour les problèmes d'optimisation à variables entières qui est la relaxation lagrangienne. Cette technique sera appliquée au GPCSTP et d'excellents résultats numériques seront présentés.

Chapitre 3

Développement de bornes inférieures basées sur la relaxation lagrangienne

3.1 Introduction

S'il existe des méthodes générales susceptibles de résoudre les problèmes d'optimisation à variables mixtes ou entières, comme les techniques de séparation et évaluation "branch–and–bound" ou la génération de coupes en programmation linéaire, ces algorithmes s'avèrent peu efficaces dès lors que le nombre de variables ou de contraintes devient élevé. Par contre, un certain nombre de problèmes, que l'on pourrait qualifier de problèmes de base, sont solutionnés rapidement et souvent élégamment par des méthodes spécifiques.

Utiliser les algorithmes efficaces imaginés pour les problèmes de base pour en résoudre de plus difficiles en éliminant de manière intelligente les contraintes supplémentaires se présente donc comme un moyen naturel de résolution. C'est dans les années 70 que s'est développée, puis popularisée, la relaxation lagrangienne où on transfère dans l'objectif, par dualisation, les contraintes compliquantes pour n'avoir plus à résoudre qu'une succession de problèmes de base, différents seulement par la fonction objectif. Les applications furent nombreuses et variées [109] [145], et il serait impossible de toutes les citer. Sur le plan théorique, il faut tout de même retenir les articles de Geoffrion [80] et Fisher [69].

Ce chapitre est dédié au développement de bornes inférieures pour le GPCSTP. Nous commençons par reformuler le GPCSTP comme un problème particulier d'optimisation d'arbre couvrant. Ensuite, nous présentons la relaxation lagrangienne ainsi que ses principaux résultats. Nous dévéloppons quatre relaxations différentes du problèmes que nous comparons théoriqement leurs performances respectives. Pour résoudre le problème dual lagrangien associé à chaque relaxation, nous appliquons treize méthodes différentes: neuf variantes de l'algorithme du sous-gradient, l'algorithme du volume, deux variantes de la méthode VTVM et une méthode exacte de génération de contraintes avec stabilisation. A travers une étude expérimentale, nous présentons une comparaison des différentes méthodes de résolution du dual lagrangien ainsi qu'une évaluation des relaxations lagrangiennes proposées.

3.2 Formulation d'arbre avec contraintes supplémentaires MSTF

En 1989, Beasley [15] a proposé une reformulation du problème de l'arbre de Steiner comme un problème d'arbre couvrant de poids minimal avec des contraintes supplémentaires. Cette reformulation a été exploitée plus tard par Lucena [145] en 1993.

Nous proposons une généralisation de cette reformulation afin de l'appliquer au GPCSTP. Nous commençons par introduire une transformation du graphe, ensuite nous redéfinissons le problème et nous présentons la formulation mathématique correspondante. En plus, une preuve de l'équivalence des deux définitions du GPCSTP sera développée.

Transformation du graphe

Partant du graphe initial $G = (V, E)$, nous construisons le graphe $\bar{G}$= $(\bar{V}, \bar{E})$ en ajoutant un noeud ficitif s et des arêtes fictives $\{s, j\}$, $\forall\ j \in V$. Ainsi, le nouveau graphe $\bar{G}$ est défini comme suit:

- L'ensemble des sommets est $\bar{V}$= $V \cup \{s\}$.
- L'ensemble des arêtes est $\bar{E}$= $E \cup E_s$, avec $E_s = \{\{s, j\}, \forall j \in V\}$. $\forall j \in V^*$, le coût de l'arrête $\{s, j\}$ est $c_{s,j} = \gamma_j$ et $c_{s,0} = 0$.

La figure ci-dessous illustre cette transformation du graphe:

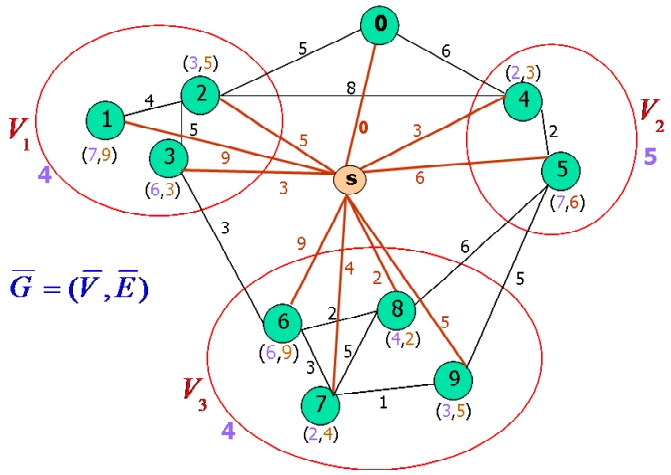

Figure 3.1: Exemple de transformation du graphe pour formuler le PCSTP comme un problème de MST avec contraintes supplémentaires

Redéfinition du problème

Avec cette transformation du graphe, le problème revient à chercher un arbre couvrant de poids minimal sur le graphe $\bar{G}= (\bar{V}, \bar{E})$. En effet, considérons un arbre couvrant $T = (\bar{V}, \bar{E}(T))$ de $\bar{G}$ vérifiant les propriétés suivantes:

(P4) Chaque sommet $j \in V^*$ adjacent à s dans T a un degré égal à 1;

(P5) La somme des profits des sommets du sous-ensemble V_k, $k = 1, ..., K$, adjacents à s est inférieure ou égale à $\bar{Q}_k = \sum_{j \in V_k} p_j - Q_k$;

(P6) $\{s, 0\} \in \bar{E}(T)$.

Théorème 19 *Il existe une bijection entre chaque arbre couvrant de $\bar{G}$ satisfaisant les propriétés (P4)-P(6) et chaque solution du GPCSTP.*

Preuve. Soit $T = (\bar{V}, \bar{E}(T))$ un arbre couvrant de $\bar{G}$ satisfaisant les propriétés *(P4)-P(6)* et ayant un poids $w(T)$. Donc, $w(T) = \sum_{e \in \bar{E}(T) \setminus E} c_e + \sum_{e \in \bar{E}(T) \cap E}$

c_e.
Soit $C = \{j \in V^* : j$ est adjacent à s dans $T\}$. En raison de $(P4)$ et comme $c_{\{s,0\}} = 0$ et $c_{\{s,j\}} = \gamma_j$, $\forall\ j \in V^*$, on a $\sum_{e \in \bar{E}(T) \setminus E} c_e = \sum_{j \in C} \gamma_j$.
Par conséquent, on obtient $w(T) = \sum_{j \in C} \gamma_j + \sum_{e \in \bar{E}(T) \cap E} c_e$.
En plus, comme tous les noeuds appartenant à C sont des feuilles de l'arbre T, alors le sous-ensemble des arêtes $\bar{E}(T) \cap E$ définit un sous-arbre $\bar{T} = (V \setminus C, \bar{E}(T) \cap E)$ qui couvre le noeud 0.
En plus, en raison de *P(5),* on a $\sum_{j \in C \cap V_k} p_j \leq \sum_{j \in V_k} p_j - Q_k\ \ \forall\ k = 1, ...K$. Ainsi, $\sum_{j \in (V \setminus C) \cap V_k} p_j \geq Q_k$, $\forall\ k = 1, ...K$.
Comme le sous-arbre $\bar{T}$ satisfait les deux propriétés *(P1)* et *(P2),* $\bar{T}$ représente une solution réalisable du PCSTP ayant un poids total égal à $\sum_{e \in \bar{E}(T) \cap E} c_e + \sum_{j \in C}$ γ_j. Ce poids est exactement le poids de l'arbre T. Et par conséqent, chaque arbre couvrant de $\bar{G}$ induit une solution réalisable du PCSTP ayant le même poids.

Inversement, considérons une solution réalisable du PCSTP, on peut facilement construire l'arbre couvrant correspondant sur le graphe élargi avec le même poids et en satisfaisant les propriétés *(P4)-(P6).* ■

Formulation mathématique

Suite à cette redéfinition du GPCSTP, une formulation mathématique en nombre entier peut être obtenue en considérant *les variables de décision* qui sont les composantes du vecteurs $x \equiv (x_{\{i,j\}})_{\{i,j\} \in \bar{E}}$ défini de la façon suivante:

$$\begin{aligned} x_{\{i,j\}} &= 1 \text{ si l'arête } \{i,j\} \in \bar{E} \text{ est couverte par l'arbre optimal} \\ &= 0 \text{ sinon} \end{aligned}$$

Par conséquent, on associe à chaque arbre réalisable un vecteur incident $x \in B^{|\bar{E}|}$, où $B = \{0, 1\}$) et ses composantes sont $x_{\{i,j\}}$ ($\{i,j\} \in \bar{E}$. Ainsi, la formulation mathématique du problème est la suivante:

$$\textbf{GPCSTP}: \text{ Minimiser } \sum_{\{i,j\} \in \bar{E}} c_{\{i,j\}} x_{\{i,j\}} \qquad (1)$$

sous contraintes:

$$\begin{aligned}
\sum_{j \in V_k} p_j x_{\{s,j\}} &\leq \bar{Q}_k, \qquad \forall k = 1, ..., K, && (2)\\
x_{\{s,j\}} + x_{\{i,j\}} &\leq 1, \qquad \forall j \in V^*,\ \{i,j\} \in E, && (3)\\
x_{\{s,0\}} &= 1, && (4)\\
x &= x_{\{i,j\}} \text{ définit un arbre couvrant de } \bar{G}, && (5)\\
x_{\{i,j\}} &\in \{0,1\}, \qquad \forall \{i,j\} \in \bar{E}. && (6)
\end{aligned}$$

La fonction objectif (1) minimise le coût total de la solution. Les contraintes (2)-(4) garantissent que la solution satisfait respectivement les propriétés *(P1)-(P3)*. La contrainte (5) exige que la solution correspond à un vecteur incident d'un arbre couvrant de $\bar{G}$. Par suite, cette dernière contrainte peut être dévéloppée en l'ensemble suivant de contraintes:

$$\begin{aligned}
\sum_{\{i,j\} \in \bar{E}} x_{\{i,j\}} &= \left|\bar{V}\right| - 1, && (5.a)\\
\sum_{\{i,j\} \in \bar{E}:\{i,j\} \in S} x_{\{i,j\}} &\leq |S| - 1, \qquad \forall S \subset \bar{V},\ 2 \leq |S| \leq \left|\bar{V}\right| - 2. && (5.b)
\end{aligned}$$

La contrainte (5.a) indique qu'on doit choisir exactement $\left|\bar{V}\right| - 1$ arêtes, et les contraintes (5.b) garantissent que l'ensemble des arêtes choisies ne contient pas de cycles.

Un résultat assez connu de Magnanti et Wolsey [151] dit que les points extrêmes du polyhèdre défini par (5.a), (5.b) et (5.c) qui suit

$$x_{\{i,j\}} \in [0,1], \qquad \forall \{i,j\} \in \bar{E}. \tag{5.c}$$

sont les vecteurs d'incidence des arbres couvrants.

Cette formulation du problème servira principalement au dévéloppement de bornes inférieures basées sur la relaxation lagrangienne.

Dans la section suivante, nous commençons par introduire le principe de la ralaxation lagangienne ainsi que ses principaux résultats. Ensuite, nous dérivons quatre relaxations lagrangiennes pour le GPCSTP.

3.3 Relaxation lagrangienne

La relaxation lagrangienne est une méthode largement utilisée pour résoudre les problèmes linéaires en nombres entiers ([80], [69], [68], [187]). En effet, il s'agit d'une technique duale qui ne fournit qu'une évaluation de la valeur optimale : une borne inférieure s'il s'agit d'une minimisation ou une borne supérieure s'il s'agit d'une maximisation.

3.3.1 Définitions et principaux résultats

Nous commençons par introduire le principe de la relaxation lagrangienne. Ensuite, nous présentons la décomposition lagrangienne. Enfin, nous rappelons les principaux résultats théoriques sur la comparaison de ces deux techniques lagrangiennes.

La relaxation lagrangienne

Introduisons d'abord la notion de la relaxation d'un problème d'optimisation combinatoire. Considérons les problèmes (P) et (PR) suivants:

$$(P)\begin{cases} Z^* = \min\limits_{x} g(x) \\ x \in \Gamma' \subseteq R^n \end{cases} \qquad (PR)\begin{cases} ZR^* = \min\limits_{x} f(x) \\ x \in \Gamma \subseteq R^n \end{cases}$$

On dit que le problème (PR) est une *relaxation* du problème (P) si et seulement si $\Gamma' \subseteq \Gamma$ et $f(x) \leq g(x)\ \forall x \in \Gamma'$.

Soient Z^* et ZR^* les solutions optimales respectives de (P) et (PR). Si (PR) est une *relaxation* du problème (P) alors $ZR^* \leq Z^*$. Ainsi, ZR^* est une borne inéfrieure du problème (P).

Pour illustrer l'approche de *la relaxation lagrangienne*, considérons le problème d'optimisation combinatoire (P) formulé comme un programme linéaire en nombre entier:

$$(P)\begin{cases} Z^* = \min cx \\ Ax \leq b & (7) \\ Bx = d & (8) \\ x \in \{0,1\}^n \end{cases}$$

Où $c^t \in IR^n$, A est une matrice (p, n), $b \in IR^p$, B est une matrice (q, n) et $d \in IR^q$.

Supposons que la contrainte (7) est une contrainte "compliquante". On peut la transférer dans la fonction objectif par dualisation en notant par $u \in IR^p$ le multiplicateur associé. On définit la relaxation lagrangienne de (P) relative à la contrainte (7) et au vecteur positif ou nul u par le problème suivant:

$$(PR_{(u)}) \begin{cases} Z_R(u) = \min\ cx + u(Ax - b) \\ Bx = d \\ x \in \{0,1\}^n \end{cases}$$

u est appelé le vecteur des multiplicateurs de lagrange et $Z_R(u)$ est appelé la fonction lagrangienne.
Ainsi, u étant connu, évaluer $Z_R(u)$ revient à résoudre $(PR_{(u)})$ où la contrainte compliquante a disparue.

Le problème de minimisation $(PR_{(u)})$ est une relaxation du problème de minimisation (P) relative à la contrainte $Ax \leq b$ car toute solution réalisable pour (P) est aussi réalisable pour $(PR_{(u)})$. De plus, si x est réalisable pour (P) et pour tout u positif, nous avons :

$$cx + u(Ax - b) \leq cx$$

Par conséquent, on obtient la propriété classique de la dualité qui dit que la valeur optimale de $(PR_{(u)})$ est inférieure ou égale à celle de (P):

$$Z_R(u) \leq Z^*$$

Ainsi, lors d'une minimisation, la méthode lagrangienne ne peut fournir qu'une borne inférieure à la valeur optimale du problème. Afin d'obtenir la meilleure borne inférieure possible, on va chercher à résoudre le programme (D) suivant:

$$(D) \qquad Z_R = \max_{u \geq 0} Z_R(u)$$

Le problème (D) est appelé le dual lagrangien du problème (P).

Remarque : Soit x une solution optimale de $(PR_{(u)})$. Si x vérifie $Ax = b$, alors x est une solution optimale de (P).

La propriété d'intégralité:

On définit *la relaxation linéaire* ou la relaxation continue du problème (P) à travers le programme $(\bar{P})$ qui suit

$$(\bar{P}) \begin{cases} \bar{Z}=\min\limits_{x} cx \\ Ax \leq b \qquad (7) \\ Bx = d \qquad (8) \\ x \in [0,1]^n \end{cases}$$

La relaxation linéaire permet d'obtenir une borne inférieure de la valeur optimale du problème (P) en relachant les contraintes d'intégralité sur les variables.

Corolaire: $\bar{Z} \leq Z_R \leq Z^*$

Ce corollaire est important dans la mesure où il contribue à expliquer la popularité dont jouit la relaxation lagrangienne. En effet, $\bar{Z}$ a longtemps été utilisé comme borne inférieure pour le problème (P), en particulier dans des méthodes branch–and–bound Z_R constitue une meilleure approximation, son usage permet par exemple de limiter le nombre de branches.

Néanmoins, il existe un certain nombre de problèmes pour lesquels Z_R est égal à $\bar{Z}$. En 1974, Geoffrion [80] a permis d'identifier une classe de problèmes ayant cette caractéristique en introduisant la notion de *propriété d'intégralité.*

Définition : *La relaxation lagrangienne possède la propriété d'intégralité si et seulement si la condition suivante est réalisée:*

Les points extrêmes du polytope$\{x \in IR^n / Bx = d\}$ *sont tous des entiers de* Z*,i.e* $\{x \in IR^n / Bx = d\} = conv\{x \in [0,1]^n / Bx = d\}$.

Où $conv\{S\}$ est l'enveloppe convexe de l'ensemble S.

Le résultat le plus important de la propriété d'intégralité est le théorème suivant:

Théorème 20 *Si la relaxation lagrangienne possède la propriété d'intégralité, alors* $\bar{Z} = Z_R$.

Ainsi, si un problème possède la propriété d'intégralité, la relaxation lagrangienne ne peut pas faire mieux que la relxation linéaire classique.

Dans le paragrphe suivant, nous allons exposer une variante de la relaxation lagrangienne qui est la décomposition lagrangienne.

La décomposition lagrangienne

La décomposition lagrangienne se présente comme une forme particulière et sophistiquée de la relaxation lagrangienne. Les articles pionners sont ceux de Shepardson et Marsten [188] qui ont appliqué la décomposition lagrangienne à un problème de planification d'horaires, de Ribeiro [178] pour la détermination du plus court chemin et de Jörnsten et Nasbergal [118] pour la résolution de problème d'affectation généralisée.

Reprenons le problème (P) sous les mêmes hypothèses. La technique de décomposition lagrangienne consiste à dupliquer les variables du problème pour le transformer en un problème équivalent:

$$(P)\begin{cases} Z^* = \min\limits_{x} cx \\ Ay \leq b & (7') \\ Bx = d & (8) \\ x = y & (9) \\ x \in \{0,1\}^n, & y \in \{0,1\}^n \end{cases}$$

En relaxant la contrainte (9), et en notant u le multiplicateur de lagrange associé, on définit le problème relaxé suivant:

$$(PD_{(u)})\begin{cases} Z_D(u) = \min\limits_{x,y} cx + u(x-y) \\ Ay \leq b \\ Bx = d \\ x \in \{0,1\}^n,\ y \in \{0,1\}^n \end{cases}$$

Comme les variables x et y ne sont pas couplées, on définit les problèmes $(PxD_{(u)})$ et $(PyD_{(u)})$:

$$(PxD_{(u)})\begin{cases} Zx_D(u) = \min\limits_{x} (c+u)x \\ Bx = d \\ x \in \{0,1\}^n \end{cases}$$

$$(PyD_{(u)})\begin{cases} Zy_D(u) = \min\limits_{x} -uy \\ Ay \leq b \\ y \in \{0,1\}^n \end{cases}$$

Il est clair que:

$$Z_D(u) = Zx_D(u) + Zy_D(u)$$

Désignons par (D_D) le problème dual lagrangien:

$$(D_D) \qquad Z_D = \max_u Z_D(u)$$

En applicant directement les propriétés établies pour la relaxation, nous pouvons affirmer que:

$$\bar{Z} \leq Z_D \leq Z^*$$

On conclut que la décomposition lagrangienne est une relaxation Lagangienne appliquée à un problème transformé de manière artficielle par la duplication des variables. Cette duplication a pour but de décomposer le problème initial en une série de sous-problèmes.

Comparaison de la décomposition et de la relaxation lagrangienne

La différence essentielle entre ces deux méthodes repose sur le fait qu'avec la décomposition, on retrouve dans l'un ou dans l'autre des sous-problèmes, chacune des contraintes du problème intial. Lorsqu'on utilise la relaxation, les contraintes relaxées n'apparaîtront jamais en tant que contraintes. Ainsi, conservant plus d'informations, la décomposition devrait donner de meilleures bornes inférieures. C'est ce qui se traduit par le théorème suivant élabli par Guignard et Kim en 1987 [93] [94].

Théorème 21 $\bar{Z} \leq Z_R \leq Z_D \leq Z^*$

Ainsi, la décomposition fournit toujours une borne de qualité au moins aussi bonne que la relaxation. Lorsqu'on inscrit cette méthode dans un algorithme de branch-and-bound, cela peut permettre de diminuer considérablement le nombre de branchement.

Inversement, la décomposition nécessite des efforts supérieurs de calcul. D'abord, pour chaque multiplicateur de Lagrange $\mathbf{u}$, on doit résoudre deux sous-problèmes (un seul avec la relaxation). Ensuite, et surtout, le dual lagrangien (D_D) associé à la décomposition est de taille bien supérieure au dual (D) associé à la relaxation. En effet, le problème (D_D) est toujours de taille n égale au nombre de variables du problème initial (P) alors que le dual (D) n'est que de taille p correspondant au nombre de contraintes complicantes. En général, p est inférieur à n.

Mais, avant de se lancer dans une décomposition lagrangienne, il faudrait savoir si une simple relaxation ne donnerait pas le même résultat. Autrement dit, il est souhaitable de pouvoir reconnaitre les cas où $Z_R = Z_D$.

Propriété d'intégralité Remarquons que face au problème (P), on peut appliquer deux relaxations lagrangiennes en relaxant la contrainte $Ax \leq b$ comme nous l'avons fait ou encore la contrainte $Bx = d$. En 1987, Guignard et Kim [93] ont donné une condition suffisante pour que Z_D soit égal à la meilleure borne inférieure obtenue en relaxant l'une ou l'autre ensemble de contraintes. Cette condition repose sur *la propriété d'intégralité.*

Théorème 22 [93] *Si* $\{x \in IR^n / Ax \leq b\} = conv\{x \in [0,1]^n / Ax \leq b\}$ *ou* $\{x \in IR^n / Bx = d\} = conv\{x \in [0,1]^n / Bx = d\}$ *alors* $Z_D = \max(Z_R, Z'_R)$.

Dans le cas où aucun des deux sous-problèmes obtenus par la relaxation lagrangienne ne posséde la propriété d'intégralité, Guignard et Kim [93] conclurent le résultat suivant:

Théorème 23 [93] *Si* $\{x \in IR^n / Ax \leq b\} \neq conv\{x \in [0,1]^n / Ax \leq b\}$ *et*

$\{x \in IR^n / Bx = d\} \neq conv\{x \in [0,1]^n / Bx = d\}$ *alors* Z_D *peut être strictement supérieur à* Z_R.

Ainsi, nous avons introduit le principe des méthodes lagragiennes ainsi que ses principaux résultats. Dans les paragraphes suivants, nous allons présenter *l'application de cette méthode au GPCSTP.*

3.3.2 Relaxation 1

Nous commençons par rappeler la formulation mathématique du GPCSTP présentée dans le paragraphe 4.2.

$$\textbf{GPCSTP}: \quad \text{Minimiser} \sum_{\{i,j\} \in \bar{E}} c_{\{i,j\}} x_{\{i,j\}} \tag{1}$$

sous contraintes:

$$\sum_{j \in V_k} p_j x_{\{s,j\}} \leq \bar{Q}_k, \qquad \forall k = 1, ..., K, \tag{2}$$

$$x_{\{s,j\}} + x_{\{i,j\}} \leq 1, \qquad \forall j \in V^*, \ \{i,j\} \in E, \tag{3}$$

$$x_{\{s,0\}} = 1, \tag{4}$$

$$x \equiv (x_{i,j})_{(i,j) \in \bar{E}} \text{ est le vecteur d'incidence d'un arbre couvrant de } \bar{G}, \tag{5}$$

$$x_{\{i,j\}} \in \{0,1\}, \qquad \forall \{i,j\} \in \bar{E}. \tag{6}$$

Comme on l'a excessivement détaillée dans le paragraphe précédent, l'idée principale de la relaxation est de faire disparaitre les contraintes compliquantes de façon à ce que le problème devient beaucoup plus facile à résoudre. Mais, le grand dilemme est de choisir les bonnes contraintes à relaxer et de savoir générer les valeurs des multiplicateurs de lagrange de façon à obtenir les meilleures bornes au problème.
Ainis, une première relaxation est obtenue en relaxant les contraintes (2-3) qu'on considère complicantes.

Soient $\alpha_k \geq 0$ $(k = 1, ..., K)$ et $\mu^j_{\{i,j\}} \geq 0$ ($j \in V^*$, $\{i,j\} \in E$) les multiplicateurs de lagrange associés respectivement aux contraintes (2) et (3). On obtient le problème relaxé suivant:

$$\begin{aligned} \theta_1(\alpha,\mu) = \quad & \text{Min} \sum_{\{i,j\}\in\bar{E}} \tilde{c}_{\{i,j\}} x_{\{i,j\}} - K_1(\alpha,\mu) \\ & \text{sous contraintes:} \quad (4-6), \end{aligned} \tag{10}$$

avec,

$$\tilde{c}_{\{i,j\}} = c_{\{i,j\}} + \mu^i_{\{i,j\}} + \mu^j_{\{i,j\}}, \quad \forall \{i,j\} \in E, \tag{11}$$

$$\tilde{c}_{\{s,j\}} = c_{\{s,j\}} + \sum_{\{i,j\}\in E} \mu^j_{\{i,j\}} + \alpha_{k_j} p_j, \quad \forall\, j \in V^*, \tag{12}$$

$$\tilde{c}_{\{0,j\}} = c_{\{0,j\}} + \mu^j_{\{0,j\}}, \quad \forall\, j \in V^* \cap \delta\{0\}, \tag{13}$$

$$K_1(\alpha,\mu) = \sum_{k=1}^{K} \alpha_k \bar{Q}_k + \sum_{j\in V^*} \sum_{\{i,j\}\in E} \mu^j_{\{i,j\}}, \tag{14}$$

où $\delta\{1\} = \{j : \{1,j\} \in E\}$ et k_j est l'indice du sous-ensemble contenant $j \in V^*$.

$\forall \{i,j\} \in \bar{E}$, $\tilde{c}_{\{i,j\}}$ représente le coût réduit de l'arête $\{i,j\}$.

On peut conclure que $\theta_1(\alpha,\mu) = MST_1(\alpha,\mu) - K_1(\alpha,\mu)$, où $T_1(\alpha,\mu)$ est la valeur optimale du problème $MST_1(\alpha,\mu)$ défini comme suit:

$$\begin{aligned} MST_1(\alpha,\mu) = \quad & \text{Min} \sum_{\{i,j\}\in\bar{E}} \tilde{c}_{\{i,j\}} x_{\{i,j\}} \\ & \text{sous contraintes:} \quad (4-6). \end{aligned}$$

En fixant les multiplicateurs de Lagrange, le problème $MST_1(\alpha,\mu))$ est le problème de l'arbre couvrant de poids minimal défini sur le graphe transformé

$\bar{G}$ avec les coûts réduits $\left(\tilde{c}_{\{i,j\}}\right)_{(i,j)\in\bar{E}}$ tout en respectant la propriété (P_6) qui dit que l'arête $\{s,0\}$ appartient à la solution.

Il est clair qu'on peut résoudre ce problème dans un temps polynômial grâce à l'algorithme de Kruskal présenté au paragraphe 2.3 avec une légère modification. En effet, l'algorithme est initialisé avec une solution partielle constituée de l'arête $\{s,0\}$ au lieu de partir de l'ensemble vide.

On conclut qu'une première borne inférieure du problème $\theta_1(\alpha,\mu)$ est calculée en résolvant le problème d'arbre couvrant de poids minimal sur le graphe $\bar{G}$ muni des coûts réduits $\left(\tilde{c}_{\{i,j\}}\right)_{(i,j)\in\bar{E}}$.

3.3.3 Relaxation 2

Une caractéristique assez intéréssante de la formulation (1)-(6) est qu'elle révèle que le polyèdre du GPCSTP est l'intersection de $K+2$ polyèdres associés à trois problèmes classiques d'optimisation combinatoire à savoir

- Le problème d'arbre couvrant de poids minimal (MST) défini par (4-6),
- K problèmes de sac à dos (Knapsac Problem KP) définis par (2) et (6),
- Le problème d'ensemble stable de poids maximum (Maximum-Weight Stable Set Problem MWSP) défini par (3) et (6).

Le MWSP est défini sur un graphe biparti $H=(A\cup B,F)$, qui est obtenu comme suit. A chaque arête $\{s,j\}$, $j\in V^*$, on associe un noeud $a_j\in A$. Et, à chaque arête $\{i,j\}\in E$, on associe un noeud $b_{\{i,j\}}\in B$. Les poids des noeuds $a_j\in A$ et $b_{\{i,j\}}\in B$ correspondent respectivement à γ_j, $j\in V^*$ et $c_{\{i,j\}}$, $\{i,j\}\in E$. Une arête $(a_j,b_{\{i,k\}})\in F$ existe si et seulement si $j\in\{i,k\}$. Le MWSP revient à chercher un ensemble de sommets dans H deux à deux non-adjacents (un *stable)* avec un poids maximal. Il s'agit d'un problème NP-difficile sur un graphe quelconque. Sauf que sur quelques graphes particuliers, on peut résoudre le MWSSP en un temps polynômial tel que les graphes bipartis [115].

Pour dériver une borne inférieure pour le GPCSTP, on peut appliquer la décomposition lagrangienne au problème défini par (1-6). L'idée principale derrière la décomposition lagrangienne est de reformuler le problème en duplicant les variables, dans le but de le décomposer en un nombre de sous-problèmes ayant des structures spéciales. Les contraintes couplantes seront relaxées selon le principe des méthodes lagrangiennes. On introduit pour

chaque variable $x_{\{s,j\}}$, $j \in V^*$, deux copies notées respectivement y_j et z_j. Ensuite, on duplique chaque variable $x_{\{i,j\}}$, $\{i,j\} \in E$, par $w_{\{i,j\}}$.

Ainsi, le problème (P) s'écrit avec ces nouvelles contraintes selon la formulation mathématique suivante:

PCSTP: Minimiser $\sum_{\{i,j\}\in\bar{E}} c_{\{i,j\}} x_{\{i,j\}}$ (15)

sous contraintes:

$$z_j + w_{\{i,j\}} \leq 1, \qquad \forall\ j \in V^*,\ \{i,j\} \in E, \quad (16)$$
$$w_{\{i,j\}},\ z_j \in \{0,1\}, \qquad \forall\ j \in V^*,\ \{i,j\} \in E, \quad (17)$$
$$\sum_{j\in V_k} p_j y_j \leq \bar{Q}_k, \qquad \forall k = 1,...,K, \quad (18)$$
$$y_j \in \{0,1\}, \qquad \forall j \in V^*, \quad (19)$$
$$x_{\{s,0\}} = 1, \quad (20)$$
$$x = (x_{\{i,j\}}) \text{ définit un arbre couvrant de } \bar{G}, \quad (21)$$
$$x_{\{s,j\}} - y_j = 0, \qquad \forall j \in V^*, \quad (22)$$
$$x_{\{s,j\}} - z_j = 0, \qquad \forall j \in V^*, \quad (23)$$
$$x_{\{i,j\}} - w_{\{i,j\}} = 0, \qquad \forall \{i,j\} \in E. \quad (24)$$

Notons α_j ($j \in V^*$), β_j ($j \in V^*$) et $\delta_{\{i,j\}}$ ($\{i,j\} \in E$) les multiplicateurs de lagrange associés aux constraintes respectives (22)-(24). En relaxant ces constraintes, on obtient la fonction lagrangienne suivante:

$$\theta_2(\alpha,\beta,\delta) = \text{Min} \sum_{\{i,j\}\in\bar{E}} \tilde{c}_{\{i,j\}} x_{\{i,j\}} - \sum_{j\in V^*} \alpha_j y_j - \Big(\sum_{j\in V^*} \beta_j z_j + \sum_{\{i,j\}\in E} \delta_{\{i,j\}} w_{\{i,j\}}\Big) \quad (25)$$

sous contraintes: $(16-21)$,

où,

$$\tilde{c}_{\{i,j\}} = c_{\{i,j\}} + \delta_{\{i,j\}}, \quad \forall \{i,j\} \in E, \quad (26)$$
$$\tilde{c}_{\{s,j\}} = c_{\{s,j\}} + \alpha_j + \beta_j, \quad \forall\ j \in V^*. \quad (27)$$

On présente le problème MWSP de la façon suivante:

$$M(\beta,\delta) = \text{Max} \sum_{j\in V^*} \beta_j z_j + \sum_{\{i,j\}\in E} \delta_{\{i,j\}} w_{\{i,j\}}$$

sous contraintes: $(16-17)$.

On définit le problème $KP_k(\alpha)$, $k = 1, ..., K$, comme suit

$$\begin{aligned} &\mathbf{KP}_k(\boldsymbol{\alpha}): && S_k(\alpha) = \text{Max} \sum_{j \in V_k} \alpha_j y_j \\ &\text{sous contraintes:} \\ & && \sum_{j \in V_k} p_j y_j \leq \tilde{B}_k, \\ & && y_j \in \{0,1\}, \quad \forall j \in V_k. \end{aligned}$$

Il est évident que $K_k(\alpha)$ est un problème de type sac à dos. Comme les noeuds du graphe sont répartis sur les K sous ensembles, l'égalité suivante est vérifiée:

$$\sum_{j \in V^*} \alpha_j y_j = \sum_{k=1}^{K} \sum_{j \in V_k} \alpha_j y_j$$

Donc, on a $\theta_2(\alpha, \beta, \delta) = T(\alpha, \beta, \delta) - \sum_{k=1}^{K} S_k(\alpha) - M(\beta, \delta)$, où $T(\alpha, \beta, \delta)$ est la valeur optimale du MST sur le graphe $\bar{G}$ en considérant les coûts réduits correspondant.

Finalement, le calcul de $\theta_2(\alpha, \beta, \delta)$ revient à résoudre $K + 2$ problèmes qui sont: un MST, K KPs et un MWSP. On observe que le MST peut être résolu en $O(|\bar{E}| + |\bar{V}| \log |\bar{V}|)$ en appliquant l'algorithme de Kruskal. Chaque KP induit par un sous-ensemble V_k, $k = 1, ..., K$, peut être résolu par un algorithme pseudo-polynômial de complexité $O(|V_k| \bar{Q}_k)$. En plus, le MWSP sur un graphe biparti peut être résolu en temps polynômial en appliquant l'algorithme primal du simplex qui ne nécéssite pas plus que $O(q^2)$ étapes pivot où q est le nombre total de variables [115].

3.3.4 Relaxation 3

Une seconde alternative pour l'application de la décomposition lagrangienne est l'introduction pour chaque variable $x_{\{s,j\}}$, $j \in V^*$, une copie notée y_j. Ce qui nous ramène à la nouvelle formulation (1), (3) et (18-22). On choisit de relaxer les contraintes (3) et (22) en leurs associant ces multiplicateurs de lagrange respectivement $\mu^j_{\{i,j\}} \geq 0$, $j \in V^*$, $\{i, j\} \in E$ et α_j, $j \in V^*$.

On obtient ainsi la fonction lagrangienne suivante:

$$\theta_3(\alpha,\mu) = \text{Min} \sum_{\{i,j\}\in\bar{E}} \tilde{c}_{\{i,j\}} x_{\{i,j\}} - \sum_{j\in V^*} \alpha_j y_j - K_3(\mu) \quad (28)$$

$$\text{sous contraintes:} \quad (18)\text{-}(21),$$

où,

$$\tilde{c}_{\{i,j\}} = c_{\{i,j\}} + \mu^i_{\{i,j\}} + \mu^j_{\{i,j\}}, \quad \forall\{i,j\}\in E, \quad (29)$$

$$\tilde{c}_{\{s,j\}} = c_{\{s,j\}} + \sum_{\{i,j\}\in E} \mu^j_{\{i,j\}} + \alpha_j, \quad \forall\ j\in V^*, \quad (30)$$

$$\tilde{c}_{\{0,j\}} = c_{\{0,j\}} + \mu^j_{\{0,j\}}, \quad \forall\ j\in V^*\cap\delta\{0\}, \quad (31)$$

$$K_3(\mu) = \sum_{j\in V^*}\sum_{\{i,j\}\in E} \mu^j_{\{i,j\}}. \quad (32)$$

Ainsi, calculer $\theta_3(\alpha,\mu)$ revient à résoudre $K+1$ problèmes séparés: un problème d'arbre de poids minimal et K problèmes de type sac à dos.

3.3.5 Relaxation 4

Une troisième décomposition lagrangienne est obtenue en associant à chaque variable $x_{\{s,j\}}$, $j\in V^*$ et $x_{\{i,j\}}$, $\{i,j\}\in E$ une copie notée respectivement z_j et $w_{\{i,j\}}$. Considérons alors la formulation $(15-17)$, (2), $(20-21)$ et $(23-24)$. Notons par β_j, $\forall j\in V^*$, $\delta_{\{i,j\}}$, $\{i,j\}\in E$ et $\alpha_k\geq 0$, $\forall k=1,...,K$, les multiplicateurs de lagrange associés respectivement aux contraintes (23), (24), et (3). En relaxant les contraintes couplantes $(23-24)$ et la contrainte de type sac à dos (2), on obtient la fonction de lagrange suivante:

$$\theta_4(\beta,\delta,\alpha) = \min \sum_{\{i,j\}\in\bar{E}} \tilde{c}_{\{i,j\}} x_{\{i,j\}} - \Big(\sum_{j\in V^*} \beta_j z_j + \sum_{\{i,j\}\in\bar{E}} \delta_{\{i,j\}} w_{\{i,j\}}\Big) - K_4(\beta,\delta,\alpha) \quad (33)$$

$$\text{sous contraintes:} \quad (16-17) \text{ et } (20-21),$$

où,

$$\tilde{c}_{\{i,j\}} = c_{\{i,j\}} + \delta_{\{i,j\}}, \quad \forall\{i,j\}\in E, \quad (34)$$

$$\tilde{c}_{\{s,j\}} = c_{\{s,j\}} + \beta_j + \alpha_{k_j} p_j, \quad \forall\ j\in V^*, j\in V_{k_j}, \quad (35)$$

$$K_4(\beta,\delta,\alpha) = \sum_{k=1}^{K} \alpha_k \bar{Q}_k. \quad (36)$$

Par conséquent, $\theta_4(\beta, \delta, \alpha)$ est calculé en résolvant deux problèmes: un MST et un MWSP.

3.3.6 Comparison des relaxations lagrangiennes

Dans ce paragraphe, nous nous intéressons à la comparaison des quatre relaxations lagrangiennes de la formulation du GPCSTP développées dans les paragraphes précédents.

Pour des raisons de commodité, nous commençons par définir les polyhèdres suivants:

- $\mathcal{X}_{KP} = \{x \in [0,1]^{|\bar{E}|} : x \text{ vérifie la contrainte (2)}\}$ et $\mathcal{P}_{KP} = \mathcal{X}_{KP} \cap \{0,1\}^{|\bar{E}|}$,
- $\mathcal{X}_{MWSP} = \{x \in [0,1]^{|\bar{E}|} : x \text{ vérifie la contrainte (3)}\}$ et $\mathcal{P}_{MWSP} = \mathcal{X}_{MWSP} \cap \{0,1\}^{|\bar{E}|}$,
- $\mathcal{X}_{MST} = \{x \in [0,1]^{|\bar{E}|} : x \text{ vérifie les contraintes (4), (5.a) et (5.b)}\}$ et $\mathcal{P}_{MST} = \mathcal{X}_{MST} \cap \{0,1\}^{|\bar{E}|}$.

Notons par $conv(\mathcal{S})$ l'enveloppe convexe d'un ensemble $\mathcal{S} \subset \mathcal{R}^{|\bar{E}|}$.

Soit $c \equiv (c_{\{i,j\}})_{\{i,j\} \in \bar{E}}$, une formulation concise du PCSTP est la suivante:

$$\textbf{PCSTP}: \quad \underset{x \in conv(\mathcal{P}_{MST} \cap \mathcal{P}_{KP} \cap \mathcal{P}_{MWSP})}{Minimiser} \quad cx \tag{37}$$

On définit

$$\theta_1^* = \max_{\alpha,\mu \geq 0} \theta_1(\alpha, \mu), \qquad \theta_2^* = \max_{\alpha,\beta,\delta} \theta_2(\alpha, \beta, \delta),$$

$$\theta_3^* = \max_{\mu \geq 0, \alpha} \theta_3(\alpha, \mu) \qquad \text{et } \theta_4^* = \max_{\alpha \geq 0, \beta, \delta} \theta_4(\beta, \delta, \alpha)$$

Pour la clarté et la complétude de ce document, nous rappelons brièvement deux principaux résultats. Considérons le programme en nombre entier suivant:

$$(P): \quad \text{Min}\{cx : A_1 x \leq b_1, A_2 x \leq b_2, x \text{ entier}\}.$$

Soit $\mathcal{P}_i = \{x \text{ entier} : A_i x \leq b_i\}$ pour $i = 1, 2$.
On peut appliquer la relaxation lagrangienne en dualisant les contraintes $A_1 x \leq b_1$. On obtient ainsi le dual lagrangien suivant:

$$(\tilde{D}) : \quad \underset{u \geq 0}{\text{Max}} \ \text{Min}\{cx + u(A_1 x - b_1) : A_2 x \leq b_2, x \text{ entier}\}.$$

Considérons la relaxation suivante de (P):

$$(\tilde{P}) : \quad \text{Min}\{cx : A_1 x \leq b_1, x \in \mathcal{P}_2\}.$$

En 1974, Geoffrion [80] a montré le lemme suivant qui établit la relation entre le dual lagrangien $(\tilde{D})$ et le problème relaxé $(\tilde{P})$.

Lemme 20 [80] *la valeur optimale de* $(\tilde{D})$ *est égale à la valeur optimale de* $(\tilde{P})$.

Une conséquence directe de ce lemme est le corollaire suivant:

Corollaire 21 $\theta_1^* = \underset{x \in conv(\mathcal{P}_{MST}) \cap \mathcal{X}_{KP} \cap \mathcal{X}_{MWSP}}{Minimum} cx.$

D'autre part, il est possible de reformuler le problème (P) comme suit:

$$(P) : \quad \text{Min}\{cx : A_1 x \leq b_1, A_2 y \leq b_2, x = y, x, y \text{ entiers}\}.$$

En dualisant la contrainte $x = y$, on obtient le dual lagrangian suivant:

$$(\hat{D}) : \quad \underset{v}{\text{Max}} \ \text{Min}\{cx + v(x - y) : A_1 x \leq b_1, A_2 y \leq b_2, x, y \text{ entiers}\}.$$

Un résultat similaire à celui du lemme précédent a été établi par Guignard et Kim [93] concernant la décomposition lagrangienne. Il dit que le dual lagrangien $(\hat{D})$ est équivalent à une relaxation primale de (P).

Lemme 22 [93] *la valeur optimale de* $(\hat{D})$*est égale à la valeur optimale de*

$$(\hat{P}) : \quad Min\{cx : x \in conv(\mathcal{P}_1) \cap conv(\mathcal{P}_2)\}.$$

Ce résultat peut être facilement généralisé à plus que deux ensembles explicites de contraintes. Ainsi, une conséquence du Lemme 21 est le corollaire suivant:

Corollaire 23 $\theta_2^* = \underset{x \in conv(\mathcal{P}_{MST}) \cap conv(\mathcal{P}_{KP}) \cap conv(\mathcal{P}_{MWSP})}{Minimum} cx.$

Remarque 1 *Notons que le Lemme 21 est aussi une conséquence directe du Lemme 19 de Geoffrion [80]. En effet, comme l'ensemble des contraintes* $\{x = y, (x, y) \in conv\{(x, y)$ *entier*: $A_1 x \leq b_1, A_2 y \leq b_2\}\}$ *est aux restrictions de* $(\tilde{P})$ *du Lemme 19, il est équivalent à* $x \in conv(\mathcal{P}_1) \cap conv(\mathcal{P}_2)$ *comme le présage* $(\hat{P})$ *dans le Lemme 21.*

A présent, considérons le programme en nombre entier général suivant avec trois structures de contraintes:

$$(Q): \quad \text{Min}\{cx : A_1 x \leq b_1, A_2 x \leq b_2, A_3 x \leq b_3, x \text{ entier}\}.$$

Le problème (Q) peut être reformulé en dupliquant ses varibles de la manière suivante:

$$(Q): \quad \text{Min}\{cx : A_1 x \leq b_1, A_2 y \leq b_2, A_3 x \leq b_3, x = y, x, y \text{ entiers}\}.$$

Un application possible de la relaxation lagrangienne consiste à dualiser les deux contraintes $A_3 x \leq b_3$ et $x = y$. Ce qui nous ramène au dual lagrangien suivant:

$$(D_Q): \quad \underset{u \geq 0, v}{\text{Max}}\, \text{Min}\{cx + u(A_3 x - b_3) + v(y - x) : A_1 x \leq b_1, A_2 y \leq b_2, x, y \text{ entiers}\}.$$

En appliquant les Lemmes 19 et 21, et en tenant compte de la Remarque 1, on obtient le Lemme suivant:

Lemme 24 *La valeur optimale de* (D_Q) *est égale à la valeur optimale de*

$$(\tilde{Q}): \quad \text{Min}\{cx : A_3 x \leq b_3, x \in conv(\mathcal{P}_1) \cap conv(\mathcal{P}_2)\}.$$

Une conséquence directe de ce lemme est le corollaire qui suit.

Corollaire 25 *Les égalités suivantes sont vérifiées:*

$$\theta_3^* = \underset{x \in conv(\mathcal{P}_{MST}) \cap conv(\mathcal{P}_{KP}) \cap \mathcal{X}_{MWSP}}{Minimum} cx,$$
$$\theta_4^* = \underset{x \in conv(\mathcal{P}_{MST}) \cap \mathcal{X}_{KP} \cap conv(\mathcal{P}_{MWSP})}{Minimum} cx.$$

Corollaire 26

$$\theta_1^* = \theta_4^* \ \leq \ \theta_2^* = \theta_3^*.$$

Preuve. Comme $conv(\mathcal{P}_{MWSP}) = \mathcal{X}_{MWSP}$, alors $\theta_1^* = \theta_4^*$ et $\theta_2^* = \theta_3^*$. De même, comme $conv(\mathcal{P}_{KP}) \subset \mathcal{X}_{KP}$, alors $\theta_1^* \leq \theta_2^*$. ■

Nous rappelons que l'égalité $conv(\mathcal{P}_{MST}) = \mathcal{X}_{MST}$ traduit le fait que le problème de l'arbre couvrant de poids minimal (MST) possède la propriété d'intégralité [151]. Ainsi, θ_1 ne peut pas faire mieux que la relaxation linéaire du modèle défini par (1)-(6). Par conséquent, θ_1 et θ_4 donnent exactement les mêmes valeurs que la relaxation linéaire du modèle défini par (1)-(6).

Dans le paragrphe suivant, nous allons analyser *treize approches parmettant de résoudre le problème dual lagrangien.* Il s'agit de douze algorithmes approchés et d'une méthode exacte que nous avons appliqués pour obtenir des bornes inférieures au GPSTP.

3.4 Méthodes de résolution du dual lagrangien

Une procédure simple de résolution du problème dual lagrangien de manière approchée est *la méthode du sous-gradient classique.* D'autres variantes de l'algorithme du sous-gradient ont été développées durant ces dernières décennies. Pour résoudre le dual lagrangien, Barahona et Anbil [10] ont publié, en 2000, un nouveau algorithme qu'ils ont appelé l'algorithme de volume. Une extension plus poussée des méthodes du sous-gradient a été introduite par Lemaréchal en 1975 [142] appelée les méthodes des faisceaux. Ces méthodes visent à surmonter le problème d'instabilité numérique en s'appuiant sur la recherche de directions de montée. Les résultats de convergence sont résumés dans le texte de Lemaréchal [143], qui présente également plusieurs illustrations numériques convaincantes. L'une des méthodes exactes de résolution du dual lagrangien les plus connus est la méthode de génération de contraintes ou encoure la programmation linéaire généralisée [126].

Dans le but de faciliter la dérivation d'une borne inférieure rapide et rigoureuse, nous avons examiné plusieurs techniques d'optimisation nondifférentiable pour résoudre le problème dual lagrangien. Plus précisément, nous avons examiné neuf variantes de l'algorithme du sous-gradient, l'algorithme du volume et la méthode VTVM (Variable Target Value Method)

appliquée en conjonction avec l'algorithme du volume et avec la méthode de plans de coupe de Polyak-Kelley généralisée (Generalized Polyak-Kelley cutting plane method GPKC). Pour une résoluton exacte du dual lagrangien, nous avons aussi développé une méthode de gération de coupes avec stabilisation.

3.4.1 Méthode du sous-gradient et ses variantes

Etant donnée la fonction duale $\theta(\pi)$, où π est le vecteur des multiplicateurs de lagrange, le dual lagrangien est défini par

$$\text{LD} : \text{Maximiser } \{\theta(\pi) : \pi \in \Pi\}, \tag{38}$$

où Π impose une restriction de non-négativité sur quelques composantes bien appropriées de π.

La méthode la plus simple et la plus populaire pour résoudre le dual lagrangien est le fameux algorithme du sous-gradient. Les méthodes du sous-gradient sont proposées pour minimiser des fonctions convexes (ou maximiser des fonctions concaves) non nécéssairement différentiables pour lesquelles il est relativement aisé de déterminer un sous-gradient en un point. Dans toute procédure d'optimisation basée sur le sous-gradient, à la $k^{ième}$ itération, le sous-gradient courant g^k est utilisé pour composer la direction de recherche d^k afin de mettre à jour le vecteur des multiplicateurs de lagrange π^k.

Une description formelle de l'algorithme du sous-gradient est donnée ci-dessous.

Algorithme du sous-gradient

Etape 0: Initialisation

Choisir $\pi^0 = 0$ comme vecteur intial des multiplicateurs de lagrange et prendre $k = 0$.

Etape 1: Calcul du sous-gradient

Etant donné π^k, résoudre le problème lagrangien relaxé pour obtenir une solution x^k et son coût correspondant θ_k. Calculer le sous-gradient courant g^k. Si $g^k = 0$ (en pratique, si $\left\|g^k\right\| < 10^{-6}$), alors prendre x^k comme solution optimale et *STOP*. Sinon, aller à Etape 2.

Etape 2: Calcul de la direction de recherche

Sélectionner une direction de recherche d^k (plusieurs stratégies sont décrites ci-dessous).
Si $d^k = 0$ (en pratique, si $\left\|d^k\right\| < 10^{-6}$), alors prendre $d^k = g^k$.

Etape 3: Mise à jour des multiplicateurs de lagrangian

Prendre $\pi^{k+1} = P_{\Pi}\left(\pi^k + \lambda_k d^k\right)$ avec une longueur de pas $\lambda_k = \beta_k \frac{UB - \theta_k}{\left\|d^k\right\|^2}$, où $P_{\Pi}()$ désigne la projection sur Π (dans ce contexte, ceci revient à mettre à zéro les composantes ayant une valeur négative et qui sont restreintes à la non-negativité). UB est une borne supérieure de la valeur optimale et $\beta_k \in (0, 2]$ est un paramètre du longueur du pas.
Prendre $k \leftarrow k + 1$.

Etape 4: Test de fin

Si le critère d'arrêt est vérifié, alors STOP. Sinon, aller à Etape 1.

Conditions de convergence de l'algorithme

En 1967, Polyak [169] a démontré que la suite θ_k converge vers Z^*, la solution optimale du problème, sous les seules conditions:

$$\begin{cases} \lambda_k \to 0 & (k \to +\infty) \\ \\ \sum\limits_{k=1}^{+\infty} \lambda_k = +\infty \end{cases}$$

Cependant, sous ces seules conditions, on ne peut rien avancer sur la vitesse de la convergence de l'algorithme du sous-gradient.
Si on choisit à chaque itération k la longueur du pas λ_k selon la régle suivante:

$$\lambda_k = \beta_k \frac{UB - \theta_k}{\|d^k\|^2}$$

où le coefficient de relaxation β_k vérifie la condition $0 < \beta_k \leq 2$, alors la convergence est assurée. Held et al. [106] ont montré que que même avec un choix d'un $UB = +\infty$, la convergence de l'algorithme n'est pratiquement pas affecté. Evidemment, la condition $\lambda_k \to 0$ $(k \to +\infty)$ oblige à choisir $\beta_k \to 0^+$ $(k \to +\infty)$.

Choix du coefficient de relaxation β

Etant donnée l'abscence des résultats théoriques à ce sujet, deux règles sont les plus utilisées dans la littérature pour la mise à jour du coefficient de relaxation. Ces règles sont les suivantes:

Held et Karp [105] rapportent une expérience satisfaisante avec la régle suivante:

- Prendre $\beta_k = 2$ pendant $2n$ itérations (n est une bonne mesure de la taille du problème),
- Diviser par 2 la valeur de β_k pendant un nombre q fixé d'itérations,
- Diviser par 2 la valeur de β_k toutes les q d'itérations jusqu'à ce que les θ_k soient suffisamment petits ($< \epsilon$ fixé).

Legendre et Minoux [140] ont déterminé les coefficients β_k de façon *dynamique* en tenant compte à chaque itération, de la progression de θ_k. La règle de mise à jour est la suivante:

- Si $\theta_k > \theta_{k-1}$, alors la borne inférieure s'est améliorée donc on peut dire que le coefficient β_k est bien choisi. Donc, on utilise la même valeur du coefficient de relaxation. En effet, on prend $\beta_{k+1} = \beta_k$.
- Si $\theta_k \leq \theta_{k-1}$, alors la borne inférieure a dinimué; le coefficient de relaxation θ_k est vraisemblablement trop grand, et l'on doit diminuer sa valeur pour l'itération suivante. Donc, on prend $\theta_k = \frac{\theta_k}{2}$.

En implémentation, nous nous sommes inspirés de cette dernière règle. En effet, le paramètre du longueur du pas β_k est calculé selon la régle suivante: $\beta_0 = 2$ et β_k est divisé en deux, chaque fois que θ_k n'augmente pas pendant un nombre fixé q d'itérations consécutives (en implémentation, nous avons pris $q = 5$). A chaque fois que ce paramètre est mis à jour, le vecteur des multiplicateurs de lagrange courant ainsi que la valeur objective courante sont réinitialisés respectivement aux meilleures solution duale et valeur objective obtenues jusqu'à présent.

Concernant le critère d'arrêt, l'idéal serait que l'algorithme du sous-gradient soit arrêté lorsque $g^k = 0$, à une itération k (en implémentation, $\left\|g^k\right\| < 10^{-6}$). Malheureusement, en pratique, ceci ne se produit qu'assez rarement. Dans la littérature, on choisit généralement comme critère d'arrêt l'une des conditions suivantes:

- Lorsque θ_k n'augmente pas pendant un certain nombre fixé d'itérations.
- Lorsque$|\theta_k - UB| < \varepsilon$ où ε correspond à un gap de tolérance fixé.

En implémentation, nous arrêtons l'algorithme après l'éxécution d'un nombre maximal d'itérations égale à 2000.

La direction de recherche est initialisée à $d^0 = g^0$. Selon le principe de mise à jour de d^k, plusieurs variantes de l'algorithme du sous-gradient peuvent être

définies comme décrites ci-dessous.

La variante SGA1 Cette variante est l'algorithme du sous-gradient classique intrdouit par Polyak en 1969 [170], où $d^k = g^k$ est choisi comme direction de recherche à chaque itération k. Cette strétégie a été largement analysée par Polyak ([169], [170]), Held et al. [106], Goffin [83] et Shor [200].

La méthode du sous-gradient classique SGA1 a tendance de zigzager, i.e une itération dans la direction du sous-gradient est généralement suivie par une itération dans la direction opposée. Ceci engendre une très lente convergence à la valeur optimale du problème.

La variante SGA2 Une modification de la méthode traditionnelle du sous-gradient a été proposée en 1975 par Camerini et al. [22] pour éviter de zigzager. L'idée est de comparer le sous-gradient le plus récent avec la direction précédente et de réduire les composantes du sous-gradient qui sont parallèles mais inverses à la direction précédente.

Pour cette variante, la direction de recherche est donnée par $d^k = g^k + \sigma_k^2 d^{k-1}$, où le scalaire σ_k^2 est calculé comme suit:

$$\sigma_k^2 = \begin{cases} -\gamma g^k.d^{k-1} / \left\|d^{k-1}\right\|^2 & , \text{ si } g^k.d^{k-1} < 0, \\ 0 & , \text{ sinon.} \end{cases}$$

γ est un réel quelconque de $(0, 1]$. Comme Camerini et al. [22], nous avons ont pris $\gamma = 1,5$.

Cette version de la méthode du sous-gradient a été testé sur plusieurs problèmes tel que le problème de Steiner dans les graphes [11].

La variante SGA3 Il s'agit d'une version modifiée de la variante SGA2. En effet, elle a été proposée par les mêmes auteurs [22]. La direction de recherche est donnée par $d^k = g^k + \sigma_k^{2'} d^{k-1}$ et le scalaire $\sigma_k^{2'}$ est calculé comme suit:

$$\sigma_k^{2'} = \begin{cases} \left\|g^k\right\| / \left\|d^{k-1}\right\| & , \text{ si } g^k.d^{k-1} < 0, \\ 0 & , \text{ sinon.} \end{cases}$$

La variante SGA4 De façon similaire aux variantes précédentes, Crowder [42] a proposé une direction de recherche $d^k = g^k + \sigma^3 d^{k-1}$, où σ^3 est un scalaire *fixé*. En se basant sur nos résultats expérimentaux, nous avons trouvé que la valeur $\sigma^3 = 0.8$ donne les meilleures performances.

La variante SGA5 Partant de la constatation que la méthode du sous-gradient classique est une méthode qui démarre rapidement mais qui devient lente à cause du zigzaguement, Jünger et al. [120] ont proposé de chosir comme direction de recherche une combinaison convexe du sous-gradient actuel et du sous-gradient précédent, i.e $d^k = \alpha g^k + (1-\alpha)g^{k-1}$, pour $\alpha \in (0,1]$. Parmis plusieurs valeurs possibles de α, nous avons trouvé que $\alpha = 0.7$ donne les meilleurs résultats numériques.

L'application de cette variante sur le problème de voyageur de commerce [120] a donnée de meilleurs résultats que l'algorithme du sous-gradient classique SGA1. Sauf que, Jünger et al. [120] calculent la longueur du pas selon la règle suivante:

$$\begin{cases} \lambda_0 = \text{constante}, \\ \lambda_k = \gamma \lambda_{k-1}, \end{cases}$$

où, γ est une constante dont la valeur influence énormement la convergence. Plus la valeur de γ est proche de 1, plus la convergence est rapide et plus les bornes inférieures obtenues sont meilleures.

La variante SGA6 Cette variante a été proposée par Sherali et Ulular en 1990 [199]. Comme dans le cas de SGA2, la direction de recherche est donnée par $d^k = g^k + \sigma_k^5 d^{k-1}$, sauf que le paramètre de déviation est $\sigma_k^5 = \frac{\|g^k\|}{\|d^{k-1}\|}$. Cette variante est généralement appelée la strétégie ADS (*Average Direction Search (ADS))* [193]. Elle a donné d'excellents résultats dans la génération de bornes inférieures pour la problème de localisation des facilités multiples [192].

La variante SGA7 Cette variante se base sur la méthode quasi-Newton réduite, telle qu'elle est décrite par Sherali et Ulular [198]. La direction de recherche est donnée par $d^k = g^k + \sigma_k^6 d^{k-1}$, où le paramètre de déviation est calculé comme suit

$$\sigma_k^6 = \begin{cases} 0 & \text{si } (g^k - g^{k-1}).d^{k-1} = 0 \\ -\frac{(g^k - g^{k-1} + d^{k-1}).g^k}{(g^k - g^{k-1}).d^{k-1}} & \text{sinon.} \end{cases}$$

Les variantes SGA8 et SGA9 Ces deux variantes sont appliquées dans le cas où π vérifie la contrainte de non-négativité. Supposons que la direction

de recherche est calculée par $d^k = \alpha_k g^k + (1-\alpha_k)d^{k-1}$, où $\alpha_k \in [0,1]$ est un coefficient à déterminer.

Nous avons dévéloppé cette variante comme est une stratégie de déviation optimisée. En effet, le coefficient α_k prend la valeur α^* telle que

$$\alpha^* = \arg\max_{\alpha_k \in [0,1]} \hat{\theta}(\alpha_k), \tag{39}$$

où $\hat{\theta}(\alpha_k) \equiv \theta\left[P_\Pi\left(\pi^k + \lambda_k d^k\right)\right]$ et la longueur du pas $\lambda_k = \beta_k \frac{UB-\theta_k}{\|g^k\|^2}$ est choisi de façon à ce qu'elle dépend de α_k.
Au lieu de résoudre (39) de manière exacte, nous avons opté pour la strétgie "single quadratic-fit" suivante:

Etape 1: Calculer $\hat{\theta}(1)$ et $\hat{\theta}(0.8)$. Si $\hat{\theta}(1) \geq \hat{\theta}(0.8)$, alors prendre $\alpha_k^* = 1$. Sinon, aller à Etape 2.

Etape 2: Calculer $\hat{\theta}(0)$. Si $\hat{\theta}(0) > \hat{\theta}(0.8)$, alors prendre $\alpha_k^* = 0$. Sinon, aller à Etape 3.

Etape 3: Calculer $\hat{\alpha}_k$ selon la formule suivante:

$$\hat{\alpha}_k = 0.5\left[\frac{-0.36\hat{\theta}(0) + \hat{\theta}(0.8) - 0.64\hat{\theta}(1)}{-0.2\hat{\theta}(0) + \hat{\theta}(0.8) - 0.8\hat{\theta}(1)}\right].$$

Etape 4: Calculer $\hat{\theta}(\hat{\alpha}_k)$. Si $\hat{\theta}(\hat{\alpha}_k) \geq \hat{\theta}(0.8)$, alors prendre $\alpha_k^* = \hat{\alpha}_k$. Sinon, prendre $\alpha_k^* = 0.8$ et STOP.

La variante SGA8 est ainsi décrite. En plus, en considérant la longueur du pas égale à $\lambda_k = \beta_k \frac{UB-\theta_k}{\|d^k\|^2}$ et en appliquant la même stratégie pour calculer α_k^*, on obtient la variante SGA9.

La méthode de sous-gradient a pour avantage la simplicité d'implantation et une certaine robustesse pour obtenir des solutions proches de l'optimum en un temps raisonable. Cependant, ces méthodes peuvent éventuellement ne pas converger à la solution optimale du problème dual. En effet, à part le cas trivial où le sous-gradient s'annule, il n'y a aucun moyen de prouver l'optimalité dans une optimisation par la méthode du sous-gradient. Ce qui explique le développement de nouvelles approches pour résoudre le dual lagrangien tel que l'algorithme du volume que nous présentons dans le paragraphe qui suit.

3.4.2 Algorithme du volume

Depuis les travaux de Held et Karp [105] et de Held et al. [106] dans les années 70, la méthode du sous-gradient a été utilisée pour déterminer des bornes inférieures pour plusieurs programmes d'optimisation et la littérature contient plusieurs expériences où cette méthode a donné d'excellentes approximations de la solution duale optimale [162]. Mais, l'un des inconvenients de la méthode du sous-gradient est qu'elle ne donne pas de solution primale au problème traité. Depuis les années 80, plusieurs tentatives ont été affectuées pour déterminer des solutions primales par la méthode du sous-gradient ([5], [136] et [193]).

En 1997, Barahona et Anbil ont présenté pour la première fois un nouvel algorithme qu'ils ont appellé l'*Algorithme de volume* (Volume Algotithm VA) [10]. Cet algorithme est une extension de la méthode du sous-gradient qui produit une solution duale et aussi une approximation de la solution primale du problème. Le mot volume est dû au fait que les solutions primales sont déterminées en estimant les volumes entre les facettes du problème duale. En effet, comme il s'agit d'une approximation de la solution primale, elle est dans la plus part des cas non réalisable. En 2002, Bahiense et al. [9] ont montré que l'algorithme du volume peut être considéré comme une variante de la méthode des faisceaux classique [142] et ils ont fourni deux variantes révisées de l'algorithme VA.

En notant $\bar{\pi}$ la meilleure solution duale obtenue jusqu'à la $k^{ième}$ itération, voici une description de l'algorithme du volume.

Algorithme du Volume (VA)

Etape 0: Choisir $\pi^0 = 0$ comme vecteur initial des multiplicateurs de lagrange. Résoudre le problème lagrangien relaxé pour obtenir une solution x^0 et son coût correspondant θ_0. Calculer le sous-gradient courant g^0.
Prendre $\bar{\pi} = \pi^0$, $d^0 = g^0$, $\bar{\theta} = \theta_0$ et $k = 1$.
Etape 1: Prendre $\pi^k = P_\Pi\left(\bar{\pi} + \lambda_k d^k\right)$, avec une longueur du pas λ_k calculée selon le formule (40). Résoudre le problème lagrangien relaxé avec π^k afin d'obtenir x^k et θ_k.
Etape 2: Prendre $d^k = \alpha g^k + (1-\alpha)d^{k-1}$, avec $\alpha \in (0, 1]$.
Etape 3: Si $\theta_k > \bar{\theta}$, alors mettre à jour $\bar{\theta} = \theta_k$ et $\bar{\pi} = \pi^k$.
Etape 4: Si le critère d'arrêt est vérifié, alors STOP. Sinon, prendre $k \leftarrow k+1$ et aller à Etape1.

Le critère d'arrêt de l'algorithme VA est similaire à celui décrit pour les variantes de l'algorithme du sous-gradient, i.e nous arrêtons l'algorithme au bout de 2000 itérations. Pour le calcul des paramètres λ_k et α, Bahiense et al. [8] ont proposé la formule suivante:

$$\lambda_k = \beta \frac{T - \bar{\theta}}{\|d^k\|^2}, \tag{40}$$

où, T est une valeur cible et $\beta \in (0, 2]$ est la longeur du pas. En implémentation, nous avons initialisé la valeur de T à 1.05 θ_0, ensuite chaque fois que $\bar{\theta} \geq 0.95T$, on augmente la valeur de T à $T = 1.05\bar{\theta}$.

Le coefficient de relaxation β est un réel entre 0 et 2 . La valeur prise par β dépend de la nature des itérations, qui sont de deux types:

- Une itération est dite *rouge*, chaque fois qu'il n'y a pas d'amélioration de la borne inférieure (i.e., $\theta_k < \bar{\theta}$). A la suite d'une séquence d'itérations rouges, la longueur du pas doit être diminuée.
- Une itération est dite *verte,* chaque fois que $\theta_k > \bar{\theta}$ et $d^k.g^k \geq 0$. Dans ce cas, la longueur de pas doit être augmentée.

Le coefficient de relaxation β est initialisé à la valeur $\beta = 0.1$. Après chaque itération *verte*, la valeur de β est doublée si $\beta < 1$. Après 20 itérations *rouges* consécutives, β est multipliée par 0.67, sauf si $\beta < 10^{-4}$, dans ce cas elle reste constante.

A chaque itération k, afin de choisir le paramètre de la combination convexe α, nous calculons $\alpha_{opt} = \frac{d^k.(d^k - g^k)}{\left\|d^k - g^k\right\|^2}$, ce qui revient à résoudre le problème:

$$\min_{\alpha \in R} \left\|\alpha g^k + (1 - \alpha) d^k\right\|^2 .$$

Si $\alpha_{opt} < 0$, alors on prend $\alpha = \alpha_{\max}/10$. Sinon, on choisit $\alpha = \min\{\alpha_{\max}, \alpha_{opt}\}$. La valeur $\alpha_{\max}$ est initialisée à 0.1. Ensuite, après chaque 250 itérations, si $\bar{\theta}$ a augmenté au moins de 1% et si $\alpha_{\max} \geq 10^{-5}$, alors on divise par deux la valeur $\alpha_{\max}$.

Des régles de mise à jour assez similaires ont été appliquées avec succés au problème de l'arbre de Steiner [8].

Remarque Barahona et Ladányi [11] ont proposée une classification des itérations légèrement différente de celle que nous avons adoptée. En efft, ils considérent qu'il y a les trois types d'iterations suivants:

- Une itération est dite *rouge*, chaque fois qu'il n'y a pas d'amélioration de la borne inférieure (i.e $\theta_k \leq \bar{\theta}$). Une séquence d'itérations rouges impose une diminution de la longueur du pas.

  Si $\theta_k > \bar{\theta}$, on calcule le produit scalaire $\varphi = d^k g^k$.

- Une itération est dite *jaune,* si $\varphi < 0$. Ainsi, en prenant un pas plus grand dans le sens de g^k, la valeur de la borne inférieure θ_k va diminuer.

- Une itération est dite *verte,* si $\varphi \geq 0$. Donc, on a besoin de prendre un pas plus grand.

Ainsi, la règle utilisée pour la mise à jour du coefficient de relaxation β est la suivante:

$$\begin{cases} \text{Après chaque itération verte,} & \beta \leftarrow \min(1.1\beta, 2) \\ \text{Après 2 itérations jaunes consécutives,} & \beta \leftarrow \min(1.1\beta, 2) \\ \text{Après 20 itérations rouges consécutives,} & \text{Si } \beta \geq 0.0005 \Rightarrow \beta \leftarrow 0.66\beta \end{cases}$$

Nous avons implémenté et testé cette régle [11] et nous avons constaté que la règle de Bahiense et al. [8] donne de meilleures bornes inférieures pour le GPCSTP.

Dans le paragraphe suivant, nous présentons une nouvelle méthode, aussi récente que l'algorithme du volume, qu'on appelle la méthode VTVM.

3.4.3 Méthode VTVM et ses variantes

En 2000, la méthode *VTVM (Variable Target Value Method)* a été introduite par Sherali et al. [195] pour résoudre les problèmes d'optimisation non-differentiable. Une caractéristique assez intérésante de cette méthode est qu'elle n'exige aucune connaissance préalable de la borne supérieure de la solution optimale, tout en garantissant la convergence à l'optimum. En effet, la méthode VTVM met à jour continuellement une estimation de la valeur optimale, appelée la valeur cible (*target value)*, en se basant sur les améliorations consécutives de la valeur objective.

Cette procédure peut être appliquée conjointement avec plusieurs stratégies de recherche de direction de recherche et de choix de la longueur du pas. Ainsi, la VTVM se distingue par une grande flexibilité. Lim et Sherali [144] ont proposée une amélioration de la méthode VTVM appliquée conjointement avec une dizaine de strétégies de choix de direction de recherche. Ils ont conclu que la meilleure performance est celle d'une variante de la méthode des plans de coupe de Polyak-Kelley généralisée (Generalized Polyak-Kelley cutting plane method GPKC) [194]. En plus, ils ont trouvé que la méthode VTVM appliquée conjointement avec l'algorithme du volume (VA) donne d'assez bons résultats. Motivés par les conclusions de Lim et Sherali [144], nous avons choisi de tester la performance de la méthode VTVM appliquée conjointement avec VA et GPKC. Une description formelle de la méthode VTVM et ainsi que des sous-routines VA et GPKC est la suivante.

Notation

Compteurs:
$l \equiv$ compteur des itérations des boucles extérieures,
$k \equiv$ compteur des itérations des boucles intérieures,
$\tau \equiv$ compteur des itérations des boucles courantes,
$\gamma \equiv$ compteur des itérations successives sans amélioration.

Pour chaque itération k:
$\pi^k \equiv$ vecteur des multiplicateurs de lagrange,
$x^k \equiv$ solution du problème lagrangian relaxé en π^k,
$\theta_k = \theta(\pi^k)$,
$g^k \equiv$ vecteur sous-gradient en π^k,
$d^k \equiv$ vecteur direction de recherche,
$\lambda_k \equiv$ longueur du pas,
$z_k \equiv$ meilleure valeur objective obtenue jusqu'à cette itération.

Pour chaque boucle extérieure l:
$w_l \equiv$ valeur cible,
$\varepsilon_l \equiv$ tolérance acceptable pour déclarer que la meilleure valeur objective actuele est suffisamment proche de la valeur cible w_l.

En plus, $\bar{\pi} \equiv$ meilleure solution duale, $\bar{g} \equiv$ sousgradient en $\bar{\pi}$, $\eta \equiv$ fraction des améliorations cumulées de la boucle intérieure qui est utilisée pour augmenter la valeur cible et $\Delta \equiv$ améliorations cumulées lors des iterations

en cours de la boucle intérieure.

Paramètres fixes

$\varepsilon_0 \equiv$ paramètre de terminaison pour la tolérance de la norme du sous-gradient $= 10^{-6}$,

$\varepsilon \equiv$ paramètre de terminaison pour la tolérance de la convergence globale $= 0.1$,

$k_{\max} \equiv$ nombre maximal d'iterations $= 2000$,

$\beta \in (0, 1] \equiv$ longueur du pas $= 0.8$,

$\sigma \in \left(0, \frac{1}{3}\right] \equiv$ paramètre d'acceptation pour la mise à jour de la valeur cible $= 0.15$,

$\bar{\tau} \equiv$ nombre maximal d'iterations sans aucune amélioration significative $= 75$,

$\bar{\gamma} \equiv$ nombre maximal d'iterations successives sans aucune amélioration (initialisé à 20 et incrémenté par 10 chaque fois que cette limit est atteinte, jusqu'à la valeur 50).

Etape 0 : Initialisation

Choisir une solution de départ π^1, et déterminer θ_1 et g^1.

Si $\|g^1\| \leq \varepsilon_0$, alors STOP.

Sinon, prendre $\bar{\pi} = \pi^1$, $z_1 = \theta_1$ et $\bar{g} = g^1$.

Initialiser $k = 1$, $l = 1$, $\tau = 0$, $\gamma = 0$, $\Delta = 0$ et l'indicateur $RESET = 1$.

Prendre $r = 0.1$, $\bar{r} = r + 1$ et $\eta = 0.75$.

Initialiser la valeur cible w_1 comme indiqué ci-dessous et prendre $\varepsilon_1 = \sigma(w_1 - \theta_1)$.

Etape 1 : Boucle intérieure de l'itération principale

Appeler une sous-routine qui définit une stratégie de détermination d'une direction de recherche et d'une longueur de pas afin d'obtenir π^{k+1} (sous-routine VA ou GPKC décrite ci-dessous).

Mettre à jour $\tau \leftarrow \tau + 1$, $k \leftarrow k + 1$ et $RESET = 0$.

Calculer θ_k et g^k.

Si $\theta_k > z_{k-1}$, alors aller à Etape 2a. Sinon, aller à Etape 2b.

Etape 2a : Amélioration dans la boucle intérieure

Mettre à jour les paramètres $\Delta \leftarrow \Delta + (\theta_k - z_{k-1})$, $z_k = \theta_k$, $\bar{\pi} = \pi^k$, $\bar{g} = g^k$ et $\gamma = 0$.

Si $k > k_{\max}$ ou $\left\|g^k\right\| \leq \varepsilon_0$, alors STOP. Si $z_k \geq w_l - \varepsilon_l$, alors aller à Etape 3a. Si $\tau \geq \bar{\tau}$, alors aller à Etape 3b. Sinon, retourner à Etape 1.

Etape 2b : Pas d'amélioration dans la boucle intérieure

Prendre $z_k = z_{k-1}$, $\gamma \leftarrow \gamma + 1$.

Si $k > k_{\max}$, alors STOP. Si $\tau \geq \bar{\tau}$ ou $\gamma \geq \bar{\gamma}$, alors aller à Etape 3a. Sinon, retourner à Etape 1.

Etape 3a : Boucle extérieure de l'itératin avec succès

Calculer $w_{l+1} = z_k + \max\{\varepsilon_l + \eta\Delta, r|z_k|\}$ et $\varepsilon_{l+1} = \max\{\sigma(w_{l+1} - z_k), \varepsilon\}$.

Si $k \leq 500$, alors mettre à jour $\eta \leftarrow 2\eta$. Sinon, prendre $\eta = 0.75$.

Si $r|z_k| > \varepsilon_l + \eta\Delta$, alors mettre à jour $r \leftarrow \frac{r}{\bar{r}}$. Prendre $\tau = 0$, $\Delta = 0$, $l \leftarrow l + 1$ et retourner à Etape 1.

Etape 3b : Boucle extérieure de l'itératin avec échec

Calculer $w_{l+1} = \frac{z_k + \varepsilon_l + w_l}{2}$ et $\varepsilon_{l+1} = \max\{\sigma(w_{l+1} - z_k), \varepsilon\}$.

Si $k \leq 500$, alors mettre à jour $\eta \leftarrow \max\left\{\frac{\eta}{2}, 0.75\right\}$. Sinon, prendre $\eta = 0.75$.

Si $\gamma \geq \bar{\gamma}$, alors prendre $\bar{\gamma} = \min\{\bar{\gamma} + 10, 50\}$.

Si $(w_l - w_{l+1}) \leq 0.1$, alors prendre $\beta \leftarrow \max\left\{\frac{\beta}{2}, \varepsilon_0\right\}$.

Prendre $\gamma = 0$, $\tau = 0$, $\Delta = 0$, et $l \leftarrow l + 1$. Mettre à jour $\pi^k = \bar{\pi}$, $\theta_k = z_k$, $g^k = \bar{g}$ et $RESET = 1$. Retourner à Etape 1.

Pour évaluer la valeur cible initiale w_1, on a exécuté 50 itérations de SGA6 et on a conservé la meilleure valeur objective obtenue $\bar{\theta}$ et aussi la meilleure solution duale $\bar{\pi}$. Le vecteur des multiplicateurs de lagrange a été initialisé à $\pi^1 = \bar{\pi}$ et la valeur cible a été calculée selon la régle suivante:

$$w_1 = \begin{cases} \bar{\theta} + 0.6(\bar{\theta} - \tilde{\theta}_0) & \text{si } \bar{\theta} \neq \tilde{\theta}_0, \\ \bar{\theta} + \varepsilon_0 & \text{sinon,} \end{cases}$$

avec $\tilde{\theta}_0 \equiv \theta(\tilde{\pi}_0)$ et $\tilde{\pi}_0 \equiv 0$.

Une description de la sous-routine VA est la suivante:

Sous-routine VA

Etape 1 - Si $RESET = 1$, alors choisir la direction de recherche $d^k = g^k$. Sinon, prendre $d^k = \alpha g^k + (1 - \alpha)d^{k-1}$, où α est un paramètre tel que $\beta \leq \alpha \leq 1$ (nous avons pris $\alpha = 0.8$). Si $\left\|d^k\right\| \leq \varepsilon_0$, alors prendre $d^k = g^k$.

Etape 2 - Calculer la longueur du pas $\lambda_k = \beta \frac{w_l - \theta_k}{\left\|d^k\right\|^2}$ et retourner $\pi^{k+1} = P_\Pi\left(\pi^k + \lambda_k d^k\right)$.

Voici une description de la sous-routine GPKC, telle qu'elle est proposée par Sherali et al. [194]:

Sous-routine GPKC

Etape 1 - Prendre $\delta = k - \bar{k}$ et $\delta' = \min\{\delta, 4\}$, où $\bar{k}$ est le numéro de l'itération la plus récente à la quelle le paramètre *Reset* a été mis à 1. Calculer $\hat{\theta} = \theta_k + \beta(w_l - \theta_k)$.

Etape 2 - Prendre $\Psi_1 = \frac{\hat{\theta}-\theta_k}{\|g^k\|^2}$ et $\Psi_2 = 0$. Calculer $\pi^{k+1} = \pi^k + \Psi_1 g^k$. Si $Reset = 1$, alors prendre $\bar{k} = k$ et sortir de la sous-routine avec $\pi^{k+1} = P_{\Pi}\left(\pi^{k+1}\right)$.

Etape 3 - Calculer $\zeta_1 = (\hat{\theta}-\theta_k)+(\pi^k)^T g^k$ et $\zeta_2 = (\hat{\theta}-\theta_{k-1})+(\pi^{k-1})^T g^{k-1}$. Si $\zeta_2 \leq (\pi^{k+1})^T g^{k-1}$, alors aller à Etape 6.

Etape 4 - Prendre $\Psi_2 = \frac{\zeta_2-(\pi^k)^T g^{k-1}}{\|g^{k-1}\|^2}$ et $\pi^{k+1} = \pi^k + \Psi_2 g^{k-1}$. Si $\Psi_2 > 0$ et $(\pi^{k+1})^T g^k \geq \zeta_1$, alors aller à Etape 6.

Etape 5 - Calculer $\Psi_0 = \|g^{k-1}\|^2 \|g^k\|^2 - \left[(g^{k-1})^T g^k\right]^2$. Si $\Psi_0 < \varepsilon_0$, alors prendre $\pi^{k+1} = \pi^k + \Psi_1 g^k$ et sortir de la sous-routine avec $\pi^{k+1} = P_{\Pi}\left(\pi^{k+1}\right)$. Sinon, calculer Ψ_1 et Ψ_2 comme suit:

$$\Psi_1 = \left[\|g^{k-1}\|^2 \left(\zeta_1 - (\pi^k)^T g^k\right) - \left((g^{k-1})^T g^k\right)\left(\zeta_2 - (\pi^k)^T g^{k-1}\right)\right] / \Psi_0,$$

$$\Psi_2 = \left[\|g^k\|^2 \left(\zeta_2 - (\pi^k)^T g^{k-1}\right) - \left((g^{k-1})^T g^k\right)\left(\zeta_1 - (\pi^k)^T g^k\right)\right] / \Psi_0.$$

Prendre $\pi^{k+1} = \pi^k + \Psi_1 g^k + \Psi_2 g^{k-1}$.

Etape 6 - Si $\delta' = 1$, alors sortir de la sous-routine avec $\pi^{k+1} = P_{\Pi}\left(\pi^{k+1}\right)$. Prendre $h = 2$ et aller à Etape 7.

Etape 7 - Calculer $\zeta_3 = \hat{\theta} - \theta_{k-h} + (\pi^{k-h})^T g^{k-h}$ et $\Psi_3 = \zeta_3 - (\pi^{k+1})^T g^{k-h}$. Si $\Psi_3 \leq 0$, alors aller à Etape 9.

Etape 8 - Prendre $\hat{\pi}^h = \pi^{k+1} + \frac{\Psi_3}{\|g^{k-h}\|^2} g^{k-h}$. Si $(\hat{\pi}^h)^T g^k \geq \zeta_1$ et $(\hat{\pi}^h)^T g^{k-1} \geq \zeta_2$, alors prendre $\pi^{k+1} = \hat{\pi}^h$.

Etape 9 - Si $h = \delta'$, alors sortir de la sous-routine avec $\pi^{k+1} = P_{\Pi}\left(\pi^{k+1}\right)$. Sinon, mettre à jour $h \leftarrow h + 1$ et retourner à Etape 7.

Les différentes méthodes proposées ci-dessus à savoir l'algorithme du sous-gradient, l'algorithme du volume et la méthode VTVM, résolvent toutes le dual lagrangien de manière approchée. Dans le paragraphe suivant, nous présentons une méthode exacte de résolution du Dual lagrangien.

3.4.4 Méthode exacte de génération de coupes avec stabilisation

La méhode de génération de coupes a été introduite par Kelley en 1960 [126]. Cette méthode est appliquée pour résoudre le dual lagrangien en programmation en nombres entiers ou mixtes. Comme son nom l'indique, l'algorithme de génération de coupes est une technique de résolution exacte qui se base sur le principe de restriction du nombre de contraintes (de coupes) et la génération de celles qui sont indispensables pour l'optimalité de la solution.

Pour la clarté de ce document, nous allons commencer par présenter cette méthode sous sa version classique. Un lecteur habitué peut se dispenser et passer directement au paragraphe suivant.

Algorithme de génération de coupes classique

Nous allons décrire l'algorithme de génération de coupe basique. Pour ce but, reprenons le programme linéaire en nombre entiers (P) suivant:

$$(P)\begin{cases} Z^* = \min\limits_{x} cx \\ Ax \leq b \\ Bx = d \\ x \in \{0,1\}^n \end{cases}$$

Soit $S = \{x \in R^n / Bx = d \text{ et } x \in \{0,1\}^n\}$. Donc, le problème relaxé $(PR_{(\pi)})$ s'écrit sous la forme suivante:

$$(PR_{(\pi)})\begin{cases} \theta(\pi) = \min\limits_{x} cx + \pi(Ax - b) \\ x \in S \end{cases}$$

Sous l'hypothèse que S est discret de cardinalité K, le problème dual (D) est équivalent au problème suivant:

$$(D) \quad \theta = \max_{\pi \geq 0} \theta(\pi) \Leftrightarrow \begin{cases} \max\limits_{\pi \geq 0} \left[\min\limits_{x} cx + \pi(Ax - b)\right] \\ x \in S = \{x_1, x_2, ...x_K\} \\ \pi \geq 0 \end{cases}$$

Donc, résoudre le problème dual (D) revient à résoudre le problème suivant:

$$(D)\begin{cases} \max\limits_{\pi,\eta} \eta \\ \eta \leq cx_k + \pi(Ax_k - b) \quad \forall k = 1, ..., K \\ \pi \geq 0 \end{cases}$$

Considèrons le Programme Maître Restreint (PMR) suivant:

$$(PMR)\begin{cases} \max\limits_{\pi,\eta} \eta \\ \eta \leq cx_k + \pi(Ax_k - b) \quad \forall k = 1, ..., K' << K \\ \pi \geq 0 \end{cases}$$

En résolvant (PMR), on obtient $(\bar{\eta}, \bar{\pi})$ qui est la solution optimale du problème (PMR).
Ainsi, $(\bar{\eta}, \bar{\pi})$ est une solution optimale du problème dual (D) si et seulement si:

$$\begin{aligned} & \bar{\eta} \leq cx_k + \bar{\pi}\ (Ax_k - b) \quad \forall k = 1, ..., K \\ \iff & \bar{\eta} \leq \min_{k=1,...,K} cx_k + \bar{\pi}\ (Ax_k - b) \\ \iff & \bar{\eta} \leq \theta(\bar{\pi}) \end{aligned}$$

Donc, ayant $\bar{\pi}$, on peut calculer $\theta(\bar{\pi})$ ce qui revient à résoudre le problème $(PR_{(\bar{\pi})})$. On appelle $\bar{x}$ la solution optimale du problème $(PR_{(\bar{\pi})})$.

Ayant $(\bar{\eta}, \theta(\bar{\pi}), \bar{x})$, deux cas se présentent :

- **Premier cas :** $\bar{\eta} \leq \theta(\bar{\pi})$

$\Rightarrow \bar{\eta}$ est la solution optimale du problème dual lagrangien (D).

- **Deuxième cas :** $\bar{\eta} > \theta(\bar{\pi})$

Dans ce cas, $(\bar{\eta}, \bar{\pi})$ viole la contrainte suivante:

$$C : \eta \leq c\ \bar{x} + \pi(A\ \bar{x} - b)$$

$\Rightarrow$ On ajoute la contrainte C au problème (PMR) et on réitère.

La convergence finie de la méthode de génération de colonnes est facile à établir. Elle résulte du fait que le problème (D) a un nombre fini de contraintes et que les contraintes successivement engendrées sont nécessairement toutes différentes.

Voici une description formelle de l'algorithme de génération de contraintes:

Algorithme de génération de contraintes
Etape 0 : Initialisation $K' = 0$.
Etape 1 :Résoudre le problème maître restreint (PMR) suivant:

$$(PMR)\begin{cases} \max\limits_{\pi,\eta} \eta \\ \eta \leq cx_k + \pi(Ax_k - b) \quad \forall k = 1..K' \\ \pi \geq 0 \end{cases}$$

Notons $(\bar{\eta}, \bar{\pi})$ la solution optimale du problème (PMR).
Etape 2 : Résoudre le problème $(PR_{(\bar{\pi})})$ suivant :

$$(PR_{(\bar{\pi})})\begin{cases} \theta(\bar{\pi}) = \min\limits_{x} cx + \bar{\pi}\ (Ax - b) \\ x \in S \end{cases}$$

Notons $x_{K'+1}$ la solution optimale du problème $(PR_{(\bar{\pi})})$.
Etape 3 : Test
- Si $\bar{\eta} \leq \theta(\bar{\pi})$, alors $\bar{\eta}$ est la valeur objective optimale et STOP.
-Sinon, ajouter au problème (PMR) la contrainte suivante :

$$C_{K'+1} : \eta \leq cx_{K'+1} + \pi(Ax_{K'+1} - b)$$

$K' \leftarrow K' + 1$ et aller à Etape 1.

La méthode de génération de coupes démarre généralement très lentement à moins d'utiliser un algorithme sophistiqué qui génère dès le début un ensemble de coupes efficaces, ou encore un dispositif de stabilisation tel qu'on va décrire dans le paragraphe qui suit.

Algorithme de génaration de coupes avec stabilisation

Dans ce paragraphe, nous présentons l'algorithme de génération de coupes stabilisé que nous avons appliqué pour résoudre de manière exacte le problème dual lagrangien. Pour raison de clarté, nous décrivons cet algorithme pour résoudre le dual lagrangien obtenu par la Relaxation 1. Dans ce cas, la fonction duale s'écrit comme suit

$$\theta_1(\alpha, \mu) = \underset{x \in conv(\mathcal{P}_{MST})}{\text{Minimum}} \{cx + \alpha(A_1x - b_1) + \mu(A_2x - b_2)\} \qquad (41)$$

où $A_1x \leq b_1$ et $A_2x \leq b_2$ sont les systèmes définis respectivement par (2) et (3).

Notons par x^p, $p = 1, ..., \omega$, les points extrêmes de l'ensemble $conv(\mathcal{P}_{MST})$. Ainsi, $\theta_1(\alpha, \mu)$ peut être exprimé comme suit

$$\theta_1(\alpha, \mu) = \underset{1 \leq p \leq \omega}{\text{Minimum}} \{cx^p + \alpha(A_1 x^p - b_1) + \mu(A_2 x^p - b_2)\}. \tag{42}$$

Le dual lagrangien (LD) est le problème suivant:

$$\underset{\alpha, \mu \geq 0}{\text{Maximiser}} \ \ \theta_1(\alpha, \mu). \tag{43}$$

Ce dernier problème est équivalent au programme linéaire suivant:

$$\text{Maximiser } \theta_1 \tag{44}$$

sous contraintes:

$$\theta_1 \ \leq \ cx^p + \alpha(A_1 x^p - b_1) + \mu(A_2 x^p - b_2), \quad \forall p = 1, ..., \omega, \tag{45}$$

$$\alpha, \mu \ \geq \ 0. \tag{46}$$

Ce programme linéaire implique aussi bien des variables primales que duale. On peut en dériver une formulation toute duale de la manière suivante:

Pour un point extrême donné x^p, soit $\alpha^p, \mu^p \geq 0$ deux vecteurs satisfaisant

$$\theta_1(\alpha^p, \mu^p) = cx^p + \alpha^p(A_1 x^p - b_1) + \mu^p(A_2 x^p - b_2). \tag{47}$$

Notons que le vecteur $g^p = (A_1 x^p - b_1, A_2 x^p - b_2)$ est un sous-gradient de la fonction dual $\theta_1(.)$ en (α^p, μ^p). Par suite, (44) peut être reformulée de la manière suivante:

$$\theta_1 \leq \theta_1(\alpha^p, \mu^p) + g^p(\alpha - \alpha^p, \mu - \mu^p), \qquad \forall p = 1, ..., \omega. \tag{48}$$

Ainsi, on obtient le programme linéaire défini par (44), (48) et (46) ne comprenant que des paramètres duaux. En général, ce problème contient un très grand nombre de contraintes, sauf que ce nombre est fini et que la plupart de ces contraintes sont inactives pour une solution optimale. En effet, ces contraintes ne doivent pas être toutes générées de manière exhaustive.

Au lieu d'appliquer l'algorithme de génération de contraintes classique, tel qu'il est présenté dans le paragraphe précédent, nous avons résolu le

programme linéaire en appliquant un algorithme de génération de coupes stabilisé similaire à celui récemment proposé Kallehauge et al. [121]. Cet algorithme a été appliqué pour générer, avec succès, une borne inférieure lagrangienne pour le problème de tournée de véhicules avec fenêtre de temps.

Cette version améliorée de l'algorithme de génération de coupes peut être considérée comme une extension de la méthode *Boxstep* proposée par Marsten et al. en 1975 [154]. On renvoie le lecteur au Chapitre XV du livre de Hirriart-Urruty et Lemaréchal [108] pour une étude détaillée et approfondie des méthodes de génération de coupes stabilisées.

Globalement, l'idée est d'ajouter les contraintes (48) de manière itérative. Plus précisément, supposons qu'à la $k^{ième}$ itération, k solutions duales (α^1, μ^1), (α^2, μ^2), ..., (α^k, μ^k) sont déjà générées. Notons par $(\bar{\alpha}^k, \bar{\mu}^k)$ la meilleure solution duale générée jusqu'à présent, i.e., $\theta_1(\bar{\alpha}^k, \bar{\mu}^k) = \max_{1 \leq p \leq k} \theta_1(\alpha^p, \mu^p)$. On considère que $(\bar{\alpha}^k, \bar{\mu}^k)$ joue le rôle d'un centre de stabilité.

Considérons le programme maître restreint suivant:

$$\mathbf{PMR}_{k+1} : \theta_1^{k+1} = \text{Maximiser } \theta_1 \tag{49}$$

sous contraintes:

$$\begin{aligned}
\theta_1 &\leq \theta_1(\alpha^p, \mu^p) + g^p.(\alpha - \alpha^p, \mu - \mu^p), \quad \forall p = 1, ..., k, && (50)\\
\left\|\alpha - \bar{\alpha}^k\right\|_1 &\leq \Delta_k^{\alpha}, && (51)\\
\left\|\mu - \bar{\mu}^k\right\|_1 &\leq \Delta_k^{\mu}, && (52)\\
\alpha, \mu &\geq 0. && (53)
\end{aligned}$$

Les contraintes (50)-(51) jouent le rôle d'un dispositif de stabilisation qui oblige la nouvelle solution $(\alpha^{k+1}, \mu^{k+1})$, obtenue après la résolution de $\mathbf{PMR}_{k+1}$, à être suffisamment proche du centre de stabilité par rapport à la norme $\|.\|_1$. Le vecteur des paramètres $\Delta_k = (\Delta_k^{\alpha}, \Delta_k^{\mu})$ représente la région de confiance.

Après la résolution de $\mathbf{PMR}_{k+1}$, on calcule $\delta_{k+1} = \theta_1^{k+1} - \theta_1(\bar{\alpha}^k, \bar{\mu}^k)$. Deux cas peuvent se présenter:

Premier cas: Si $\delta_{k+1} = 0$ (en pratique, $\delta_{k+1} < 10^{-5}$), alors prendre $(\bar{\alpha}^k, \bar{\mu}^k)$ comme solution optimale et STOP.

Deuxième cas: Si $\delta_{k+1} > 0$, on résoud un problème d'arbre couvrant de poids minimal afin de dériver aussi bien la valeur de $\theta_1(\alpha^{k+1}, \mu^{k+1})$ que

celui du sous-gradient g^{k+1} en $(\alpha^{k+1}, \mu^{k+1})$. Notons l'augmentation nette de la valeur objective par $\bar{\delta}_{k+1} \equiv \theta_1(\alpha^{k+1}, \mu^{k+1}) - \theta_1(\bar{\alpha}^k, \bar{\mu}^k)$. Si le ratio $\rho = \frac{\bar{\delta}_{k+1}}{\delta_{k+1}} > 0.01$, alors le centre de stabilité est mis à jour en prenant $(\bar{\alpha}^{k+1}, \bar{\mu}^{k+1}) = (\alpha^{k+1}, \mu^{k+1})$. Sinon, on prend $(\bar{\alpha}^{k+1}, \bar{\mu}^{k+1}) = (\bar{\alpha}^k, \bar{\mu}^k)$. En plus, la région de confiance est mise à jour de la manière suivante:

- Si $\rho = 1$, alors on prend $\Delta_{k+1} = 1.5\Delta_k$,
- Si $0 < \rho < 1$, alors on prend $\Delta_{k+1} = \Delta_k$,
- Si $\rho < 0$, alors on prend $\Delta_{k+1} = 0.9\Delta_k$.

Après cette mise à jour des paramètres, on ajoute au programme $\mathbf{PMR}_{k+1}$ la contrainte (50) indicée par $(\alpha^{k+1}, \mu^{k+1})$. C'est ainsi qu'on obtient le nouveau programme maître $\mathbf{PMR}_{k+2}$ qui sera résolu de nouveau comme détaillé ci-dessus.

Pour ce qui concerne l'initialisation de l'algorithme, nous avons pris $\Delta_1 = \mathbf{1}$ (i.e., un vecteur dont toutes les composantes sont égales à 1) et (α^1, μ^1) égale à la meilleure solution duale obtenue après l'exécution de l'algorithme du sous-gradient SGA6 pendant 100 itérations.

Dans la section suivante, nous présentons une étude expérimentale comparative des différentes méthodes de résolution du dual lagrangien ainsi que des différentes relaxations.

3.5 Etude expérimentale

Pour évaluer la performance des algorithmes d'optimisation non-différentiable appliqués aux différentes relaxations étudiées, nous les avons codé et compilé en utilisant le langage de programmation Microsoft Visual $C++$ (version 6.0).

Chaque algorithme a été exécuté sur un ordinateur Pentium IV 2.4 GHz, afin de résoudre les Relaxations 1 et 2. En effet, en raison du Corollaire 26, les Relaxations 3 et 4 ne peuvent pas fournir de meilleures bornes inférieures.

Les différents algorithmes proposés ont été testés sur 25 instances que avons générées aléatoirement. Les paramètres des instances sont générés comme suit:

- Les coûts des arêtes $(c_{ij})_{(i,j)\in E}$ et les profits $(p_j)_{j\in V^*}$ sont des entiers générés aléatoirement selon la distribution discrète uniforme $\pi[10, 25]$.

- Pour chaque noeud $j \in V^*$, on associe une pénalité $\gamma_j = \lceil 0.5 p_j \rceil$.
- Le budget minimum à collecter au sein de chaque sous ensemble de sommet V_k, $k = 1, ..., K$, est égal à $Q_k = \left\lceil \alpha \sum_{j \in V_k} p_j \right\rceil$ avec $\alpha = 0.5$.

En prenant $\alpha = 0.5$, nous nous intéressons au problème où le profit minimum à collecter $Q = \sum_{k=1}^{K} Q_k$ est presque égal à 50% de la somme des profits de tous les sommets. C'est une façon de chercher à connecter 50% des sommets du réseau.

Pour chaque instance, nous avons appliqué l'algorithme de génération de coupes stabilisé afin de déterminer les valeurs exactes de θ_1 et θ_2.

Les caractéristique de ces instances sont présentées dans le tableau 3.1. La signification des colonnes est comme suit:
- $Inst.$ $\equiv$ le nom de l'instance,
- $n \equiv$ le nombre de sommets,
- $m \equiv$ le nombre des arêtes,
- $K \equiv$ le nombre de sous-ensembles,
- $\theta_1^* \equiv$ la valeur optimale de θ_1,
- $\theta_2^* \equiv$ la valeur optimale de θ_2,
- $Temps_1 \equiv$ le temps CPU nécéssaire au calcul de θ_1^* en secondes,
- $Temps_2 \equiv$ le temps CPU nécéssaire au calcul de θ_2^* en secondes.

Dans les tableaux 3.2 et 3.3, on présente en sommaire la performance de tous les algorithmes d'optimisation non-différentiable que nous avons dévéloppés pour calculer respectivement θ_1 et θ_2. Pour chaque procédure et pour chaque instance, on introduit deux mesures de performance:
- $Gap_i = 100 \times \frac{\theta_i^* - \theta_i}{\theta_i^*}$, où θ_i est la borne obtenue, $i = 1, 2$,
- $Temps \equiv$ le temps CPU total en secondes.

Inst.	n	m	K	θ_1^*	θ_2^*	θ_1^* / θ_2^*	Temps$_1$	Temps$_2$
Inst.01	10	15	3	91	95	0.9579	0.15	2.23
Inst.02	15	30	5	148	156	0.9487	0.34	1.10
Inst.03	26	65	5	245	252	0.9722	0.76	15.29
Inst.04	30	75	7	286	294	0.9728	3.70	48.64
Inst.05	70	140	7	685	694	0.9870	331.28	783.96
Inst.06	40	100	10	378	395	0.9570	56.36	55.42
Inst.07	50	110	10	513	534	0.9607	22298.92	460.39
Inst.08	50	250	5	451	459	0.9826	84.26	90.42
Inst.09	60	120	15	616	642	0.9595	357.14	188.93
Inst.10	70	450	50	643	715	0.8993	641.21	245.34
Inst.11	80	120	20	873	931	0.9377	660.56	459.17
Inst.12	100	150	20	1072	1118	0.9589	48263.89	1043.04
Inst.13	100	400	5	899	907	0.9912	474.60	81.32
Inst.14	90	150	25	950	1002	0.9481	30961.29	118.93
Inst.15	90	450	30	787	843	0.9436	155.85	255.71
Inst.16	120	180	30	1280	1360	0.9412	52643.96	256.59
Inst.17	150	450	10	1445	1457	0.9921	322.46	395.92
Inst.18	175	350	35	1730	1785	0.9692	36394.84	817.34
Inst.19	160	240	40	1735	1821	0.9528	80840.25	523.48
Inst.20	180	250	45	1925	2041	0.9432	2201.21	630.64
Inst.21	190	270	45	2060	2186	0.9424	1293.18	566.79
Inst.22	200	300	50	2160	2309	0.9355	69531.12	905.61
Inst.23	200	350	50	2095	2237	0.9365	30961.29	821.48
Inst.24	250	400	75	2605	2795	0.9320	51632.96	1280.03
Inst.25	250	500	100	2630	2873	0.9154	36325.91	1296.68
Moyenne						***0.9535***		

Tableau 3.1: Caractéristiques des instances

	SGA1		SGA2		SGA3		SGA4		SGA5		SGA6		SGA7		VA		VTVM-VA		VTVM-GPKC	
Inst.	Gap1	Temps	Gap1	Temps	Gap1	Temps	Gap1	Temps	Gap1	Temps	Gap1	Temps	Gap1	Temps	Gap1	Temps	Gap1	Temps	Gap1	Temps
Inst.01	13.19	0.30	**1.10**	0.28	**1.10**	0.91	3.30	1.11	20.88	0.23	2.20	0.22	**1.10**	0.28	9.89	0.31	**1.10**	0.27	**1.10**	0.25
Inst.02	8.11	0.34	**0.68**	0.47	1.35	0.44	**0.68**	0.39	5.41	0.38	1.35	0.45	**0.68**	0.42	7.43	0.59	**0.68**	0.47	**0.68**	0.49
Inst.03	3.27	1.33	**0.41**	1.27	1.22	1.09	1.22	1.20	0.82	1.28	2.04	1.25	1.22	8.53	4.08	1.54	**0.41**	1.52	**0.41**	1.52
Inst.04	11.54	1.16	2.80	1.28	2.80	1.30	11.89	0.97	1.75	1.38	**0.70**	1.31	1.75	1.33	7.34	1.53	**0.70**	1.44	**0.70**	1.45
Inst.05	12.26	2.78	**0.44**	3.09	2.48	3.30	1.90	3.19	12.55	2.66	2.19	3.25	1.31	3.31	5.26	3.47	**0.44**	3.00	**0.44**	3.02
Inst.06	14.29	1.31	1.85	1.64	3.17	1.69	3.97	1.69	7.67	1.66	1.85	1.72	0.53	1.64	8.99	1.87	**1.06**	1.88	**1.06**	1.86
Inst.07	19.30	1.67	**0.58**	2.64	1.56	2.58	19.49	1.83	19.69	1.72	3.90	2.58	3.51	2.52	9.75	2.77	3.51	2.63	3.51	2.69
Inst.08	6.65	2.84	**0.44**	3.31	0.89	3.26	**0.44**	3.34	2.00	3.43	0.89	3.29	1.55	3.44	5.99	2.73	0.67	3.46	0.89	3.39
Inst.09	12.82	2.58	1.30	2.86	2.27	2.75	17.37	2.33	14.77	2.58	2.11	2.83	3.73	2.69	7.64	2.75	**1.14**	2.75	**1.14**	2.78
Inst.10	11.51	6.67	4.67	6.81	1.24	7.21	**0.16**	7.29	15.86	6.85	**0.16**	7.46	0.31	7.32	10.58	6.14	0.78	7.56	0.78	7.78
Inst.11	14.66	3.41	0.80	3.86	**0.11**	3.86	1.37	3.80	19.70	3.30	0.23	3.83	0.80	3.83	14.09	3.94	5.50	3.83	5.61	3.80
Inst.12	13.81	4.25	0.28	5.00	1.21	4.99	17.91	4.11	19.87	5.09	**0.09**	5.02	0.93	4.72	8.15	5.44	7.56	4.47	7.65	4.45
Inst.13	7.45	8.06	**0.11**	8.26	1.00	8.65	4.12	8.00	7.45	7.98	0.89	8.82	1.00	8.73	5.12	6.59	**0.44**	8.07	0.56	8.37
Inst.14	17.16	3.84	0.74	4.16	**0.63**	4.05	19.47	3.95	22.32	3.91	0.84	4.16	0.53	4.13	14.16	4.11	1.26	3.97	1.16	3.97
Inst.15	8.26	6.89	5.21	7.86	3.18	8.17	3.56	8.51	1.27	8.06	**0.64**	8.35	3.05	8.25	3.81	6.98	0.89	8.15	0.89	8.39
Inst.16	16.72	5.59	2.97	6.08	1.48	5.94	0.63	6.00	20.78	4.97	**0.86**	5.95	1.56	5.80	7.35	5.94	5.08	5.94	5.16	6.00
Inst.17	9.55	10.68	0.42	12.46	0.90	13.17	2.77	13.32	9.55	11.81	2.91	13.79	3.37	13.26	4.98	9.34	2.15	12.17	2.15	12.42
Inst.18	15.03	10.88	2.77	11.75	**0.46**	11.77	15.49	11.53	16.99	10.53	**0.46**	11.88	0.19	11.94	8.64	12.63	3.47	11.33	3.53	11.47
Inst.19	17.46	9.09	2.88	9.00	**2.07**	9.27	0.86	9.17	21.79	7.64	2.31	9.23	2.71	9.42	12.38	9.70	5.36	8.88	5.48	8.91
Inst.20	16.73	10.09	2.08	10.39	0.26	10.58	0.73	10.05	4.52	10.50	**0.16**	10.66	1.35	10.63	10.80	11.11	6.39	10.41	7.27	10.53
Inst.21	15.87	11.31	3.11	11.44	0.73	11.63	1.21	11.06	2.77	11.06	**0.49**	11.45	1.75	11.56	9.92	11.98	6.84	11.34	7.14	11.55
Inst.22	17.87	12.39	2.55	12.70	1.94	13.02	9.17	11.89	3.24	12.64	**0.51**	12.86	0.60	13.19	10.94	13.25	7.73	11.89	7.92	11.86
Inst.23	14.13	15.47	0.67	14.84	0.38	15.09	17.18	13.27	2.34	14.52	**0.29**	14.86	0.62	15.20	5.69	14.92	3.63	14.92	3.48	14.75
Inst.24	17.01	19.39	1.15	20.38	0.69	20.19	21.04	18.16	20.15	18.50	**0.65**	19.98	0.15	20.08	9.53	19.91	5.64	18.98	5.80	18.58
Inst.25	17.53	23.41	0.80	24.61	0.99	24.41	22.62	21.55	21.79	22.00	**0.15**	24.30	0.49	24.27	8.91	25.25	5.70	24.03	5.82	23.69
Moyenne	*13.29*		*1.63*		*1.36*		*7.94*		*11.84*		*1.15*		*1.39*		*8.46*		*3.13*		*3.21*	
Ecart Type	*4.14*		*1.40*		*0.88*		*8.15*		*8.15*		*1.01*		*1.07*		*2.85*		*2.58*		*2.68*	

Tableau 3.2: Performance des algorithmes non-différentiables par rapport à la Relaxation 1

	SGA1		SGA2		SGA3		SGA4		SGA5		SGA6		SGA7		SGA8		SGA9		VA		VTVM-VA		VTV
Inst.	Gap2	Temps	Gap2	Temps	Gap2	Temps	Gap2	Temps	Gap2	Temps	Gap2	Temps	Gap2	Temps	Gap2	Temps	Gap2	Temps	Gap2	Temps	Gap2	Temps	Gap2
Inst.01	4.21	6.19	2.11	6.05	1.05	9.28	2.11	5.56	2.11	6.67	2.11	5.52	2.11	5.36	2.11	16.49	2.11	15.27	3.16	5.36	**0.00**	5.81	**0.00**
Inst.02	7.05	6.06	**0.00**	6.74	**0.00**	10.45	0.64	6.53	1.92	7.17	**0.00**	6.23	0.64	6.41	3.85	21.91	0.00	21.08	3.82	6.258	**0.00**	6.81	0.64
Inst.03	3.97	9.44	0.79	10.27	**0.40**	9.64	0.79	10.64	1.98	10.42	**0.40**	9.70	0.40	10.14	3.17	35.91	**0.40**	39.63	2.38	10.72	**0.40**	9.64	**0.40**
Inst.04	3.06	10.08	1.02	12.66	1.36	10.53	1.36	11.42	4.42	11.56	1.02	10.50	2.72	11.03	3.06	33.33	1.02	39.36	3.74	12.46	**0.68**	11.39	1.36
Inst.05	6.34	16.50	0.29	19.22	**0.14**	18.86	0.58	18.09	5.91	19.20	**0.14**	20.49	4.18	18.11	1.87	60.58	0.43	70.14	2.02	21.79	0.29	19.31	0.43
Inst.06	11.65	13.03	1.01	15.22	0.76	13.28	1.27	14.27	4.5 6	15.61	0.76	14.02	**0.51**	13.92	7.85	40.08	**0.51**	45.56	6.09	15.35	**0.51**	13.74	**0.51**
Inst.07	5.06	15.44	1.87	15.89	1.69	15.33	**1.50**	15.44	2.62	18.11	**1.50**	15.23	3.00	15.50	5.43	52.88	1.69	64.44	5.81	17.23	1.69	15.78	1.69
Inst.08	1.96	9.95	2.61	11.03	0.87	13.42	1.53	11.05	5.23	12.27	**0.65**	15.02	2.61	10.88	1.96	36.47	1.09	53.45	2.61	32.38	1.09	12.22	0.87
Inst.09	2.80	16.42	1.25	17.64	1.25	16.97	1.56	17.39	2.18	18.75	1.25	16.47	1.71	17.23	4.67	57.45	1.25	70.95	4.67	19.54	**1.09**	17.45	1.40
Inst.10	2.38	17.48	1.54	18.50	**1.40**	20.63	1.68	18.39	6.99	20.05	**1.40**	19.36	3.22	17.88	3.22	61.53	**1.40**	86.95	2.10	35.61	**1.40**	21.84	1.54
Inst.11	4.51	17.98	2.69	19.22	**2.04**	19.47	2.15	18.70	5.26	20.19	**2.04**	19.11	4.08	18.48	6.44	61.75	2.69	80.66	4.22	21.41	3.11	17.94	2.47
Inst.12	3.94	23.56	**0.72**	24.70	**0.72**	26.55	**0.72**	24.94	1.07	25.08	**0.72**	23.88	1.79	24.48	5.10	79.00	1.07	98.84	4.50	26.48	0.89	23.53	0.81
Inst.13	3.09	18.66	2.09	21.34	**0.33**	21.77	0.55	21.39	1.65	19.02	**0.33**	23.88	2.21	19.47	2.21	75.02	0.44	104.38	1.87	76.92	0.44	21.56	0.44
Inst.14	3.39	22.11	0.70	22.95	0.40	26.91	**0.30**	24.53	1.60	24.88	0.40	23.52	0.80	24.02	4.59	77.55	0.40	95.83	4.90	25.12	0.40	23.77	0.50
Inst.15	15.71	19.69	1.56	21.11	1.20	25.13	1.32	21.33	2.52	23.88	**1.20**	21.92	1.56	21.64	3.84	72.97	1.32	98.89	5.64	60.84	1.92	26.03	1.32
Inst.16	4.85	29.27	1.25	29.05	1.03	32.88	1.99	27.70	2.65	28.92	1.10	30.05	1.62	30.91	3.75	95.17	**0.96**	119.16	8.17	31.49	1.03	27.39	1.03
Inst.17	7.28	25.20	0.27	25.42	0.21	29.44	1.17	26.59	5.49	27.08	0.21	27.98	2.13	25.28	1.37	99.66	**0.07**	142.36	1.17	95.08	0.21	33.14	0.21
Inst.18	1.06	45.39	0.11	47.66	**0.06**	47.34	**0.06**	45.66	0.89	48.09	**0.06**	48.69	1.23	52.95	3.86	169.97	0.28	196.41	6.31	54.13	0.11	47.13	0.62
Inst.19	10.76	35.19	0.55	37.20	0.66	38.58	**0.38**	35.58	2.03	36.30	**0.38**	36.03	1.04	42.59	5.60	122.59	0.66	154.56	3.25	41.09	0.66	34.77	1.04
Inst.20	2.25	39.58	0.98	41.14	**0.78**	39.47	**0.78**	40.06	1.76	40.38	**0.78**	42.39	1.03	48.30	3.72	121.45	1.47	170.44	7.21	45.23	1.22	39.88	0.83
Inst.21	3.98	42.36	0.96	41.94	1.05	42.95	**0.82**	43.31	1.33	44.95	**0.82**	49.92	1.37	52.94	3.66	147.45	1.19	184.53	5.49	48.22	1.28	43.88	1.19
Inst.22	9.48	44.64	1.56	46.33	1.43	47.42	1.47	48.17	1.56	51.63	**1.26**	51.89	1.52	54.05	4.94	160.89	1.52	199.38	7.42	52.57	1.52	47.36	1.65
Inst.23	5.19	48.38	1.16	47.98	1.03	49.33	2.01	49.52	2.95	50.59	**0.89**	55.20	1.39	59.56	4.38	214.47	0.98	221.34	3.11	57.16	1.21	50.69	1.92
Inst.24	11.59	63.86	1.68	60.09	1.18	60.95	1.25	61.25	3.15	63.23	**1.07**	68.19	1.25	67.55	4.97	216.30	1.54	261.95	7.02	68.1	1.50	61.27	1.82
Inst.25	7.14	74.80	0.84	70.22	0.77	70.83	**0.70**	73.89	3.62	71.17	0.73	73.88	1.98	80.23	4.56	236.49	0.84	306.97	4.59	78.26	**0.70**	71.48	0.80
Moyenne	*5.71*		*1.18*		*0.87*		*1.15*		*2.95*		*0.85*		*1.84*		*4.01*		*1.01*		*4.45*		*0.93*		*1.02*
Ecart Type	*3.63*		*0.72*		*0.52*		*0.59*		*1.70*		*0.56*		*1.01*		*1.51*		*0.64*		*1.94*		*0.71*		*0.61*

Tableau 3.3: Performance des algorithmes non différentiables par rapport à la Relaxation 2

A partir du tableau 3.1, on observe que l'algorithme exacte de génération de coupes stabilisé peut résoudre des instances de grande taille tout en ne nécéssitant que des temps de convergence assez raisonable. En plus, on constate que les bornes obtenues par la décomposition lagrangienne sont nettement supérieures aux bornes obtenues par la relaxation lagrangienne.

En examinant les tableaux 3.2 et 3.3, on peut faire les observations suivantes:

- SGA6 (la stratégie ADS) donne d'excellents résultats qui sont nettement meilleurs que ceux de toutes les autres méthodes. En effet, cet algorithme fournit des bornes inférieures avec un gap moyen de 1.15% et 0.85% par rapport aux valeurs optimales respectives des Relaxations 1 et 2. En plus, il manifeste une robustesse de performance à travers un écart type moyen assez faible.

- Malgrès sa simplicité, SGA3 montre une bonne performance pour les deux relaxations. SGA7 et VTVM-VA donnent d'assez bons résultats aussi.

- SGA9 est systématiquement plus performant que SGA8. Ceci peut être expliqué par un choix plus judicieux de la direction de recherche dans la formule de la longueur du pas. Cependant, ces deux algorithmes sont très couteux en terme de temps de calcul.

- Les résultats de l'algorithme du Volume sont assez décevantes en comparaison de ceux des différents algorithmes du sous-gradient dévié.

- Ce n'est pas étonnant que l'algorithme du sous-gradient classique (SGA1) soit vraiment à la traine de toutes les autres méthodes d'optimization non-différentiable.

- En terme d'effort de calcul et de rapidité de convergence, la Relaxation 2 nécéssite beaucoup plus de temps de calcul que la Relaxation 1.

3.6 Conclusion

Ce chapitre a été conscré à l'application des outils de l'optimisation non-différentiable et en particulier à la relaxation lagrangienne qui a été extensivement appliquée sur plusieurs problèmes d'optimisation combinatoire

tels que le problème du voyageur de commerce [13], le problème de localisation avec contrainte de capacité [38], le problème localisation des facilités avec contrainte de capacité [81], le problème d'affectation généralisé [181], etc...
Les méthodes lagrangiennes ont été également étudiées dans le cadre des problèmes à variables entières mais à fonction objectif non-linéaire par Michelon et Maculan[157] et Wah et Wu [204].

Dans ce chapitre, nous avons présenté la relaxation et la décomposition lagrangienne ainsi que leurs principaux résultats théoriques. Nous avons aussi exposé les principales méthodes de résolution du dual lagrangien ainsi que leurs avantages et leurs inconvenients.
Une première formulation mathématique a été proposée ainsi que quatre différentes relaxations lagrangiennes. Une analyse théorique des bornes inférieures correspondantes a été développée ainsi qu'une étude comparative d'une grande variété d'approches de résolution de problèmes d'optimisation non différentiable.

Les résultats obtenus sont très encourageants et ont fait l'objet d'un article scientifique intitulé: "*The Prize Collecting Steiner Tree Problem: Models and lagrangian Dual Optimization Approaches*" qui a été publié en 2008 dans la revue internationale "*Computational Optimization and Applications*" [102].

Dans les chapitres suivants, nous allons nous intérésser au cas particulier du GPCSTP où $K = 1$ (K est le nombre de sous-ensembles de sommets).

Chapitre 4

Développement d'une heuristique basée sur la décomposition

4.1 Introduction

Pour le reste de cette thèse, nous nous intéressons au problème de Steiner avec collecte de prix (Prize Collecting Steiner Tree Problem PCSTP). Il suffit de considérer le GPCSTP avec un seul sous-ensemble de sommets (i.e $K = 1$). Nous rappelons les données du PCSTP comme suit.

Soit un graphe non orienté $G = (V, E)$ où l'ensemble de sommets est $V = \{0, 1, ..., n\}$ et E est l'ensemble des arêtes du graphe G. En plus, 0 est un noeud particulier qu'on appelle le noeud racine. Supposons qu'à chaque sommet i de l'ensemble $V^* = V \backslash \{0\}$, on associe deux réels positifs p_i et γ_i. En effet, p_i est le profit associé au sommet i s'il est connecté au réseau en question, alors que γ_i correspond à une pénalité à payer si le sommet i n'est pas connecté au réseau. La somme des profits des sommets connectés doit être au moins égale à un profit minimum à collecter Q. Le but du PCSTP est de trouver un sous-ensemble de sommets S avec $\{0\} \subset S \subseteq V$ tel que la somme du poids de l'arbre de poids minimal couvrant S dans le graphe G et des pénalités des noeuds n'appartenant pas à l'arbre, soit minimale.

Ce chapitre est consacré au dévéloppement de bornes supérieures pour le PCSTP. L'idée de base est de construire une solution heuristique en résolvant à l'optimum successivement deux problèmes d'optimisation combinatoire très

bien étudiés dans le littérature. Dans une première étape, un arbre couvrant de poids minimal sur le graphe G est calculé sur la base des coûts des arêtes $(c_e)_{e\in E}$ (la phase *spanning*). Manifestement, cet arbre est une solution réalisable du PCSTP. Cependant, une solution améliorée peut être obtenue après avoir tailler quelques branches de l'arbre tout en le conservant réalisable au PCSTP (la phase *pruning*). Ainsi, une seconde étape consiste à résoudre un PCSTP sur l'arbre couvrant dérivé. Dans la suite, nous appelons cette approche l'*heuristique span-and-prune* (en bref HSP).

Nous présentons deux versions de cette heuristique (HSP et HSPI) qui dépendent du choix de l'arbre couvrant de la première phase.

4.2 Complexité du PCSTP défini sur un arbre

Le PCSTP est déjà connu comme un problème $\mathcal{NP}$-difficile. Nous démontrons un résultat de complexité encore plus précis.

Proposition 18 *Le PCSTP sur un graphe sous forme d'arbre est $\mathcal{NP}$-difficile.*

Preuve. Cette preuve se base sur la réduction du problème du "sac à dos binaire" (Binary Knapsack Problem, BKP) qui est $\mathcal{NP}$-difficile et dont la forme de minimisation est la suivante:

$$\text{Minimiser } \left\{ \sum_{j=1}^{n} \pi_j x_j : \sum_{j=1}^{n} a_j x_j \geq b,\ x_j \ \in \{0,1\},\ \forall j = 1,...,n \right\} \tag{54}$$

avec $\pi_j \geq 0$ et $a_j \geq 0, \forall j = 1,...,n$.

Commençons par construire un arbre $\tilde{G} = (\tilde{V}, \tilde{E})$ qui a la topologie d'un graphe étoile dont le noeud central est le noeud 0 et les feuilles sont les noeuds $1, 2, ..., n$. Ainsi, l'ensemble des arêtes est $\tilde{E} = \{\{0,1\}, \{0,2\}, ..., \{0,n\}\}$.

On définit un exemple du PCSTP sur $\tilde{G}$ comme suit. On considère le noeud 0 comme le noeud racine et on attribue un coût π_j à chaque arête $\{0,j\} \in \tilde{E}$. Pour chaque noeud feuille $j \in \tilde{V} \setminus \{0\}$, on associe un profit égale à a_j et une pénalité nulle. On fixe comme quota la quantité b.

Il est clair que chercher une solution optimale pour le BKP revient à résoudre le PCSTP sur l'arbre $\tilde{G}$ et vice versa. ■

4.3 Reformulation du problème

Malgré que ce dernier résultat de complexité semble mettre à l'épreuve la performance de toute heuristique proposée, on montre qu'il est toujours possible de résoudre un PCSTP, sur un graphe sous forme d'arbre, de façon assez efficace en appliquant un algorithme pseudo-polynômial.

Dans ce but, Notons par $T = (V, E(T))$ l'arbre couvrant de poids minimal du graphe G, et notons par $c(T)$ le coût de cette solution réalisable du PCSTP. Pour des raisons de commodité, on convertit l'arbre T en une arborescence $\vec{T}$ dont la racine est le noeud 0 et qui est obtenue comme suit.

Pour chaque noeud $j \in V^* = V \setminus \{0\}$, considérons l'arête $\{i, j\} \in E(T)$ incidente à j dans le chemin $\mathcal{P}_j$ de T entre le noeud racine et le noeud j. Ainsi, l'arête $\{i, j\}$ est remplacée par l'arc $\delta_j^- \equiv (i, j)$.
On introduit la notation suivante:

- $\bar{Q} = \sum_{j \in V^*} p_j - Q$;
- σ_j = l'ensemble des noeuds successeurs au noeud j dans $\vec{T}$, $\forall j \in V^*$;
- $a_j = p_j + \sum_{k \in \sigma_j} p_k$, $\forall j \in V^*$;
- $\omega_j = \gamma_j + \sum_{k \in \sigma_j} \gamma_k$, $\forall j \in V^*$;
- π_j = poids total de la sous-arborescence de $\vec{T}$ dont la racine est le noeud j au quel on ajoute le poids de l'arc δ_j^-, $\forall j \in V^*$.

On observe que si un arc δ_j^-, $j \in V^*$, est enlevé de $\vec{T}$ ainsi que la sous-arborescence dont la racine est le noeud j, alors on obtient une solution du PCSTP dont le coût total est $c(T) - \pi_j + \omega_j$. Cependant, cette solution est réalisable si $a_j \leq \bar{Q}$. Ainsi, résoudre le PCSTP défini sur l'arbre T revient à résoudre le problème de sac à dos avec contrainte de précédence suivant:

$$\text{Maximiser} \sum_{j \in V^*} (\pi_j - \omega_j) x_j \tag{55}$$

sous contraintes:

$$\sum_{j=1}^{n} a_j x_j \leq \bar{Q}, \tag{56}$$

$$x_i + x_j \leq 1, \quad \forall\, i \in \sigma_j,\ j \in V^*, \tag{57}$$

$$x_j \in \{0,1\},\ \forall\, j \in V^*, \tag{58}$$

où $x_j = 1$ si l'arborescence $\vec{T}$ est taillé en enlevant l'arc δ_j^-, $j \in V^*$, sinon $x_j = 0$.

La fonction objectif (55) maximise la contribution totale des arcs enlevés. Notons que le coût de la solution finale réalisable obtenue pour le PCSTP est $c(T) - \sum_{j \in V^*} (\pi_j - \omega_j) x_j$.

La contrainte (56) guarantie que la solution respecte la contrainte du quota minimal à collecter. Les contraintes (57) expriment le fait que si i est un successeur de j dans $\vec{T}$ alors on ne peut pas enlever les deux arcs δ_i^- et δ_j^-. Ceci évite de multiplier le comptage des poids des noeuds et des arêtes enlevés. Les contraintes (58) traduisent le fait que les variables de décision sont binaires.

Proposition 19 *Le problème défini par (55)-(58) peut être résolu en un temps pseudo-polynômial $O(n\bar{Q})$.*

Preuve. D'abord, on associe à l'arborescence $\vec{T}$ une permutation de son ensemble de sommets $\eta = (\eta(1), \eta(2), ..., \eta(n))$ qui manifeste la propriété du pré-ordre d'arrangement: Les sommets successeurs de chaque sommet j, $j \in V^*$, sont arrangés de manière à apparaître immédiatement après lui [2]. Plus explicitement, ceci revient à renuméroter les sommets selon le principe suivante: Si le sommet i est un prédecesseur de j alors $\eta(i) < \eta(j)$. Il est évident que $\eta(1) = 1$.
Cette permutation η peut être construite en un temps $O(n)$ en appliquant l'algorithme "*depth-first search*" (voir le livre de Ahuja et al. [2] p.73-76).

Dans la suite, on replace l'indice de chaque noeud j par son ordre correspondant $\eta(j)$. On définit le graphe orienté acyclique $D = (\hat{V}, \hat{A})$ comme suit.
L'ensemble des noeuds est $\hat{V} = V \cup \{n+1\}$. L'ensemble des arcs $\hat{A}$ est construit de la manière suivante:

- Il existe un arc $(0, j)$ pour tout $j = 1, ..., n + 1$;
- Il existe un arc $(j, n + 1)$ pour tout $j = 1, ..., n$;
- Il existe un arc (i, j) pour tout $i < j$ tel que j n'est pas un successeur de i, $\forall i, j \in V^*$.

A chaque arc $(i, j) \in \hat{A}$, on associe deux paramètres: une longueur c_{ij} et une demande d_{ij}. La longueur de tout arc incident au noeud j, $j \in V^*$, est $\pi_j - \omega_j$ et sa demande correspondante est a_j. La longueur et la demande de chaque arc incident au noeud fictif $n + 1$ sont nulles.

Appelons $\mathcal{P} = (1, j_1, ..., j_q, n+1)$ le chemin orienté du graphe D partant du noeud 0 au noeud $n+1$. Ce chemin a une longueur totale $l(\mathcal{P}) = \sum_{h=1}^{q} (\pi_{j_h} - \omega_{j_h})$ et une demande totale $d(\mathcal{P}) = \sum_{h=1}^{q} a_{j_h}$.

Ainsi, on associe à $\mathcal{P}$ la solution du système (56)-(58) définie par $x_j = 1$, $\forall j \in \{j_1, ..., j_q\}$ et $x_j = 0$, $\forall j \in V^* \backslash \mathcal{P}$. On peut déjà facilement vérifier si $d(\mathcal{P}) \leq \bar{Q}$. Dans ce cas, cette solution est nécessairement réalisable et admet un coût total égal à $l(\mathcal{P})$.

Inversement, étant donnée une solution réalisable du système (56)-(58), on peut lui associer un chemin orienté entre les noeuds 0 et $n + 1$ ayant le même coût et dont la demande totale ne dépasse pas $\bar{Q}$.
Par conséquent, on peut résoudre le problème définie par (55)-(58) de manière équivalente en résolvant le problème du plus long chemin avec une contrainte de capacité (PLCC) dans un graphe orienté acyclique. Ce dernier problème peut être résolu avec un algorithme de programmation dynamique en un temps $O(n\bar{Q})$ [117]. ■

Il est clair que visiter un noeud i sur le graphe orienté acyclique $D = (\hat{V}, \hat{A})$ revient à éliminer l'arête incidente à ce noeud i dans l'arbre initial $T = (V, E(T))$, ce qui revient à éliminer tout le sous-arbre T_i incident à i et ne contenant pas la racine 0.

Finalement, chercher une solution heuristique au PCSTP revient à résoudre un problème du plus long chemin avec une contrainte de capacité (PLCC) dans un graphe orienté acyclique. Il s'agit de l'heuristique HSP.

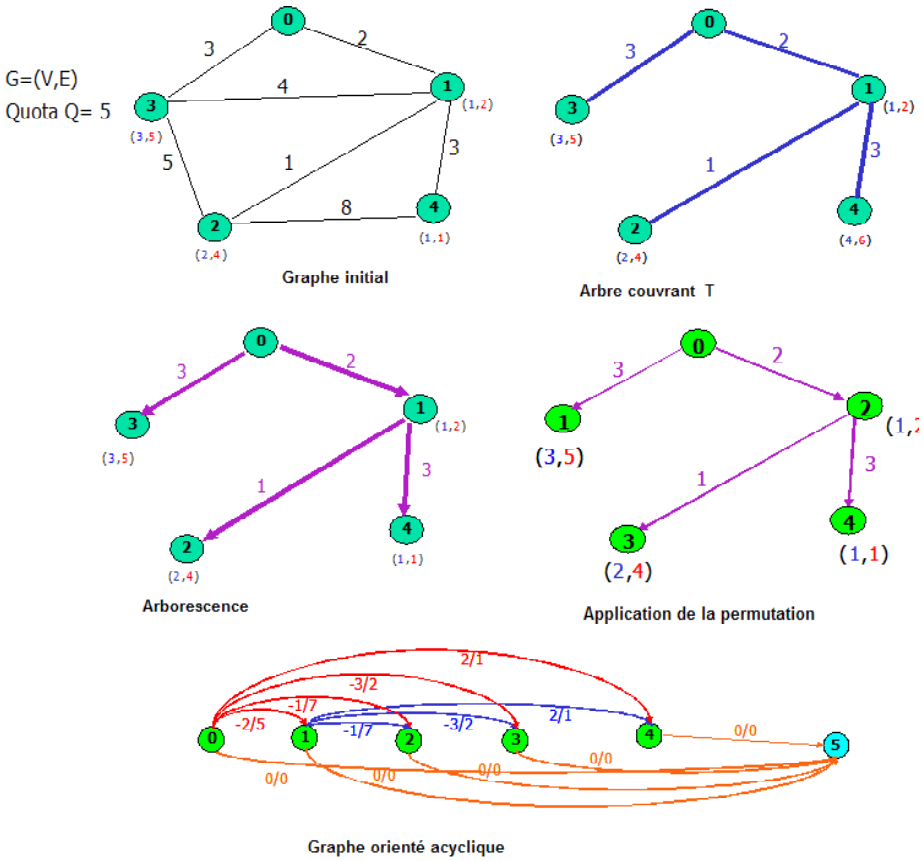

Figure 4.1: Exemple de transformation de graphe lors de l'heuristique **HSP**

4.4 Résolution du PLCC

Le problème du plus long chemin avec une contrainte de capacité (PLCC) est un problème NP-difficile. En effet, Handler et Zang [98] ont déjà montré ce résultat pour le problème du plus court chemin avec une contrainte de capacité (PCCC). Ce problème a été largement étudié lors de ces dernières décennies et ses applications sont multiples telque le problème des tournées de véhicules [104] etc...

Nous commençons par présenter une formulation mathématique du PLCC.

Formulation mathématique du PLCC

Reprenons le graphe orienté acyclique $D = (\hat{V}, \hat{A})$. A chaque arc $(i,j) \in \hat{A}$, on associe deux réels: une longueur c_{ij} et une demande d_{ij}. Considérons comme variables de décision les composantes du vecteur $x \equiv (x_{ij})_{(i,j)\in\hat{A}}$ définies comme suit.

$$\begin{aligned} x_{ij} &= 1, \text{ si le sommet } i \text{ est connecté au sommet } j, \\ &= 0, \text{ sinon.} \end{aligned}$$

La valeur x_{ij} peut être aussi interprétée comme la valeur du flot circulant sur l'arc (i,j), $(i,j) \in \hat{A}$. Ainsi, une formulation mathématique du PLCC est la suivante:

$$\text{Maximiser} \sum_{(i,j)\in\hat{A}} c_{ij}x_{ij} \tag{59}$$

sous contraintes:

$$\sum_{j\in\delta^+(i)} x_{ij} - \sum_{j\in\delta^-(i)} x_{ji} = \begin{cases} 1 & , \text{ si } i = 0, \\ 0 & , \forall i \in V^*, \\ -1 & , \text{ si } i = n+1, \end{cases} \tag{60}$$

$$\sum_{(i,j)\in\hat{A}} d_{ij}x_{ij} \leq \bar{Q}, \tag{61}$$

$$x_{ij} \in \{0,1\}\ ,\ \forall (i,j) \in \hat{A}. \tag{62}$$

Grâce aux contraintes (60) et (62), le vecteur $\mathbf{x}$ représente le vecteur d'incidence d'un chemin élémentaire entre la source 0 et le puits $n+1$. Dans la contrainte (61), le nombre positif d_{ij} représente la demande du noeud j, $j \in V^*$, et la quantité $\bar{Q}$ représente une capacité maximale sur le chemin

recherché. Cette contrainte représente une contrainte de type sac à dos; i.e une contrainte de capacité sur le chemin recherché.

Méthode exacte de résolution du PLCC

Deux approches ont été proposées pour résoudre le PCCC. La première méthode se base sur la programmation dynamique par une généralisation des algorithmes de Ford [117] et Dijkstra [159]. La deuxième approche se base sur l'application de la relaxation lagrangienne. Elle était proposée pour la première fois par Minoux en 1975 [159] pour obtenir une solution approchée du PCCC. Handler et Zang [98] ont montré comment l'améliorer pour obtenir une solution optimale.

Nous décrivons cette deuxième approche telle que nous l'avons appliquée pour résoudre le PLCC.

Etape 0: *Application de la relaxation lagrangienne*

On dualise la contrainte de capacité (61), en lui associant le multiplicateur de lagrange positif λ. Appelons $\mathcal{L}$: l'ensemble des chemins entre la source 0 et le puit $n+1$.

Ainsi obtenu, le problème dual lagrangien (D_{PLCC}) revient à chercher λ^* minimisant la fonction de lagrange suivante:

$$(D_{PLCC}) \quad \min_{\lambda \geq 0} \mathrm{L}(\lambda) = \max_{x \in \mathcal{L}} (cx + \lambda(dx - \tilde{Q})).$$

Où $c \equiv (c_{ij})_{(i,j)\in \hat{A}}$ et $d \equiv (d_{ij})_{(i,j)\in \hat{A}}$ représentent respectivement les vecteurs coût et demande.

Pour un réel positif λ donné, déterminer $L(\lambda)$ revient à chercher le plus long chemin (PLC) avec les coûts modifiés $(c + \lambda d)$. On note $c(\lambda)$ et $d(\lambda)$ respectivement le coût total et la demande totale du PLC obtenu avec les coûts modifiés $(c + \lambda d)$.
Ainsi, on obtient l'égalité $L(\lambda) = c(\lambda) + \lambda(d(\lambda) - \tilde{Q})$.

Etape 1: *Détermination de la solution duale optimale λ^* à l'aide du principe de la dichotomie.*

Etape1.0: Prendre $\lambda_1 = 0$ et $\lambda_2 = \infty$, ce qui correspond au chemin avec la demande totale minimale. Déterminer $L(\lambda_1)$ et $L(\lambda_2)$.

Etape1.1: Calculer $\lambda' = \frac{c(\lambda_1)-c(\lambda_2)}{d(\lambda_2)-d(\lambda_1)}$.
Etape1.2: Déterminer $L(\lambda')$.
Etape1.3: Si $L(\lambda') = c(\lambda_1) + \lambda'(d(\lambda_1) - \tilde{Q}) = c(\lambda_2) + \lambda'(d(\lambda_2) - \tilde{Q})$, alors $\lambda^* = \lambda'$. Sinon, si $d(\lambda') > \tilde{Q}$, prendre $\lambda_1 = \lambda'$ et aller à Etape1.1. Si $d(\lambda') < \tilde{Q}$, prendre $\lambda_2 = \lambda'$ et aller à Etape1.1.

Ayant la solution duale optimale λ^*, notons $\bar{x}^*$ la solution primale correspondante. La solution $\bar{x}^*$ est réalisable mais non nécessairement optimale. On obtient ainsi l'encadrement suivant de la valeur optimale z^* du PLCC:

$$L(\lambda^*) \leq z^* \leq c(\lambda_2)$$

En effet, $L(\lambda^*)$ correspend à une borne inférieure résultant de l'application de la relaxation lagrangienne. Et, $c(\lambda_2)$ correspend à une borne supérieure car le multiplicateur de lagrange λ_2 est associé à une solution réalisable.

Il est possible de trouver la solution optimale x^* en utilisant une *procédure de classement* proposée par Handler et Zang [98].

Etape 2: *Détermination de la solution primale optimale x^* à l'aide de la procédure de classement.*
Notons B le coût de la meilleure solution primale réalisable obtenue durant l'Etape1.2. Appelons x_g le vecteur qui correspond au $g^{ième}$ plus long chemin avec les coûts modifiés $(c + \lambda^* d)$, $g \in IN^* \backslash \{1\}$. La fonction duale L_g en ce point s'écrit:

$$L_g(\lambda^*) = cx_g + \lambda^*(dx_g - \tilde{Q})$$

L'idée de la procédure de classement est de générer au fur et à mesure les chemins $x_2, ..., x_g$ et de s'arrêter dès qu'on trouve x_g tel que $L_g(\lambda^*) > B$.

Décrivons brièvement la procédure de classement [98]:

Soit $k = 2$. Générer x_k et calculer $L_k(\lambda^*)$.
Tant que $(L_k(\lambda^*) \leq B)$ faire
    $k = k + 1$.
    Générer x_k.
    Calculer $L_k(\lambda^*)$.
    Mettre à jour B.
Fin Tant que.

La solution primale optimale est la dernière solution réalisable ayant servi à la dernière mise à jour de B.

L'apport incontestable de l'application de la relaxation lagrangienne se manifeste à travers une dimunition considérable du nombre des plus longs chemins à déterminer avant la convergence de la procédure de classement. En effet, on aurait pu chercher diretement le $k^{ième}$ PLC qui vérifie la contrainte de capacité, ce qui revient à chercher parmi une centaine de chemins. Alors qu'avec l'application de la relaxation lagrangienne, cette procédure converge au bout d'une dizaine d'itérations au maximum [104].

Pour la clarté et l'intégrité de ce rapport, nous exposons brièvement les algorithmes que nous avons appliqués pour la recherche du PLC et du $k^{ème}$ PLC.

Comme il s'agit d'un graphe acyclique orienté, on détermine le PLC en appliquant l'algorithme de Roy (1959) dévéloppé à la base pour déterminer le plus court chemin sur un graphe acyclique et dont la description est la suivante:

Algorithme de recherche de PLC pour un graphe acyclique
Notation

$C(i) \equiv$ longueur du plus long chemin entre le sommet 0 et le sommet i, $i = 1, ..., n+1$, avec $C(0) = 0$,

$\Gamma^{-1}(i) \equiv$ ensemble des prédécesseurs du sommet i, $i = 1, ..., n+1$.

Etape 0: Classer et rénuméroter les sommets par rang croissant, $i = 0, ..., n+1$.

Etape 1: Prendre $i = 1$.

Tant que $i \leq n+1$,

faire $C(i) = \max\limits_{j \in \Gamma^{-1}(i)} \{C(j) + c_{ji}\}$ et $i \leftarrow i+1$,

Fin Tant que.

La longueur du PLC entre les sommets 0 et $n+1$ est donnée par $C(n+1)$.

Pour trouver le $k^{ième}$ PLC, on applique l'algorithme de Dreyfus datant de 1968 [48]. Il est basée sur la programmation dynamique. Une description formelle de cet algorithme est la suivante:

Algorithme de Dreyfus
Notation

$P_l(j) \equiv$ le $l^{ème}$ PLC entre la racine 0 et le sommet j.
$Z(P_l(j)) \equiv$ longueur du $P_l(j)$.
$n(i,j,l) \equiv$ nombre de chemins contenant l'arc (i,j) parmi $P_1(j), P_2(j), ..., P_l(j)$.
Etape 0: Calculer de $P_0(j)$, $\forall j \in V^*$, et $n(i,j,0)$, $\forall (i,j) \in \hat{A}$.
Etape 1: Prendre $l = 1$.
Tant que $l \leq k$,
faire $Z(P_l(j)) = \max_{i \in \Gamma^{-1}(j)} \left(Z(P_{n(i,j,l-1)}(i)) + c_{ij} \right)$, $\forall j \in V^*$,

$$\begin{aligned} n(i,j,l) &= n(i,j,l-1) \quad , \text{ si } (i,j) \in P_l(j), \\ &= n(i,j,l) \qquad , \text{ sinon,} \end{aligned}$$

$l \leftarrow l + 1$,
Fin Tant que.

Ainsi, nous avons détaillé les différents algorithmes formant l'heuristique HSP. Dans le paragraphe suivant, nous développons un résultat théorique de la limite de la performance de HSP dans le pire des cas (worst-case).

4.5 Etude de la performance du pire cas

Comme présenté dans le paragraphe 4.2, le PCSTP sur un arbre peut être résolu en un temps pseudo-polynômial. Par conséquent, l'heuristique HSP donne une soluion approchée en un temps pseudo-polynômial.

Dans ce paragraphe, nous nous intéréssons à l'analyse du pire cas (*worst-case)* de la performance de l'heuristique HSP.

Proposition 20 *HSP posséde une garantie de performance non bornée*

Preuve. Considérons l'instance géométrique suivante:
Soient $1, 2, ..., n$ les sommets d'un n-polygone régulier. La longueur de chaque arête est égale à 1 unité. Pertubons légèrement ce polygone en replaçant son sommet n à une distance égale à $1 + e$, $e > 0$, par rapport à ses deux sommets voisins 1 et $n - 1$. Ainsi, la distance entre le sommet n et les sommets j, $j = 2, ..., n-2$, est strictement supérieure à $1+\epsilon$. Un profit $p_j = 1$ est associé à chaque sommet j, $j = 2, ..., n-1$, avec $p_n = n - 2$. Toutes les pénalités sont nulles et le quota est fixée à $n - 2$.
La figure ci-dessus présente le cas $n = 5$.

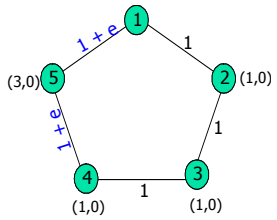

Figure 4.2: Exemple illustrant le pire cas de l'heuristique span-and-prun

Durant la première étape de HSP, on détermine l'arbre couvrant de poids minimal qui n'est d'autre que le chemin $(1, 2, ..., n)$ dont la longueur totale est $Z_{MST} = n - 1 + \epsilon$. Durant la deuxième phase, cet arbre est taillé en enlevant l'arc $(n-1, n)$ ce qui correspond à une solution réalisable du PCSTP avec un coût total $Z_{HSP} = n - 2$.

Cependant, la solution optimale inclut l'arête $\{1, n\}$, collecte un profit totale égale à $n-2$ et admet comme coût $Z^* = 1 + e$. Dans ce cas, on trouve que $\lim_{n\to\infty} \frac{Z_{HSP}}{Z^*} = +\infty$. ■

Nous constatons dans le paragraphe 4.7, où les résultats numériques sont discutés, que ce résultat négatif n'empêche pas l'efficacité pratique de l'heuristique HSP puisqu'elle donne souvent des solutions très proches de l'optimum.

4.6 Variantes de l'heuristique

Pour obtenir une borne supérieure efficace du PCSTP, nous avons développé deux versions de l'heuristique selon le choix de l'arbre couvrant de la première phase de recouvrement.

4.6.1 Heuristique HSP

Il s'agit de la version basique de l'heuristique telle que nous avons présenté dans le paragraphe 4.1. Nous rappelons qu'il suffit de prendre comme arbre de départ, sur lequel on va résoudre un PCSTP, l'arbre couvrant de poids minimal (MST) du graphe G avec les coûts réels des arêtes $(c_e)_{e\in E}$. Dans ce cas, il suffit d'appliquer l'algorithme de Kruskal.

4.6.2 Heuristique HSPI

Comme la solution finale produite par l'heuristique *span-and-prune* dépend fortement de l'arbre couvrant de la première phase, nous avons jugé utile de réitérer cette heuristique plusieurs fois avec differents arbres couvrants initiaux. Plus précisément, nous avons implémenté la stratégie itérative suivante:

Durant le calcul de bornes inférieures lagrangiennes à travers l'application de la Relaxation 2 (paragraphe 3.3.3), un arbre couvrant de poids minimal est déterminé à chaque itération de l'algorithme de sous-gradient SGA6 (paragraphe 3.4.1). Ainsi, toutes les cinquante iterations, l'arbre couvrant de poids minimal retrouvé avec les coûts réduits $(\tilde{c}_{ij})_{\{i,j\}\in E}$ est considéré comme un nouveau input pour l'heuristique span-and-prune et un PCSTP est résolu sur cet arbre. Finalement, on ne retient que la meilleure solution trouvée lors de cette procédure que nous appelons l'heuristique *span-and-prun* itérative (en bref HSPI).

Pour éviter de considérer les même arbres couvrants, nous avons fait appel à la fonction $H(T) = \sum_{i,j\in V:\{i,j\}\in E(T)} i * j * c_{ij}$ où $T = (V, E(T))$ est un arbre couvrant de G. Ainsi, pour tout T, T' arbres couvrants de G, si $H(T) = H(T')$, alors on ne retient qu'un seul arbre pour la deuxième phase de l'heuristique HSPI.

4.7 Etude expérimentale

Nous avons codé les deux versions de l'heuristique dévéloppée avec le language de programmation Microsoft Visual C++ (Version 6.0) et nous les avons implémentées et exécutées sur un ordinateur Pentium IV 3.2 GHz possédant une RAM de 3.0 GB. Comme présenté dans le paragraphe 4.4, le PLCC a été résolu par l'approche basée sur la relaxation lagrangienne décrite par Handler et Zang [98].

Pour évaluer la performance des heuristiques proposées, nous les avons testées sur trois classes d'instances benchmark de *SteinLib* [131], une bibliothèque pour le problème de l'arbre de Steiner dans les graphes. Ces classes du *SteinLib* étaient introduites par Beasley depuis 1989 [16]. Elles sont définies sur des graphes non orientés et très peu denses et ils sont générées aléaoirement avec des coûts euclidiens.
Les classes tests sont notées B, C, et D comme dans [131]. Elles sont com-

posées de 43 instances qui se différencient principalement par leurs tailles respectives. Le nombres de noeuds et des arêtes varient respectivement de 50 à 1000 et de 63 à 2500. Les profits $(p_j)_{j\in V^*}$ sont des entiers générés aléatoirement selon la distribution discrète uniforme $U[10, 25]$. Pour chaque noeud $j \in V^*$, on associe une pénalité $\gamma_j = \lceil 0.5p_j \rceil$. Pour chaque instance, le quota minimal à collecter est $Q = \left\lceil 0.5 \sum_{j\in V\setminus\{1\}} p_j \right\rceil$.

En notant $n \equiv$ nombre de noeuds et $m \equiv$ nombre d'arêtes, les instances des classes du *SteinLib* [131], que nous considérons pour cette étude expérimentale, se présentent comme suit:

- Toutes des instances de la classe B qui compte 18 instances dont la taille varie de $n = 50$, $m = 63$ à $n = 100$, $m = 200$;
- 15 instances de la classe C dont la taille varie de $n = 500$, $m = 625$ à $n = 500$, $m = 2500$;
- 10 instances de la classe D dont la taille varie de $n = 1000$, $m = 1250$ à $n = 1000$, $m = 2000$.

Les résultats obtenus sont présentés dans les tableaux 4.1-4.3. La signification des colonnes est comme suit:
- $Inst. \equiv$ le nom de l'instance,
- $n \equiv$ le nombre de noeuds,
- $m \equiv$ le nombre des arêtes,
- $Z_{HSP} \equiv$ la valeur de la solution de l'heuristique HSP,
- $Z_{HSPI} =$ la valeur de la solution de l'heuristique HSPI,
- $Z^* \equiv$ La valeur de la solution optimale (voir chapitre 5),
- $Gap\ ISPH\ = 100 \times \frac{Z_{HSPI}-Z^*}{Z^*}$,
- $Gap\ SPH\ = 100 \times \frac{Z_{HSP}-Z^*}{Z^*}$,
- $T_{HSP} \equiv$ le temps CPU nécéssaire pour obtenir Z_{HSP} en secondes,
- $T_{ISPH} \equiv$ le temps CPU nécéssaire pour obtenir Z_{ISPH} en secondes.

Inst.	n	m	Z_{SPH}	Z_{ISPH}	Z*	Gap HSP	Gap HSPI	T_{SPH}	T_{ISPH}
B.01	50	63	142	142	140	1.43	1.43	0.01	0.01
B.02	50	63	175	153	153	14.38	0.00	0*	0*
B.03	50	63	130	128	128	1.56	0.00	0*	0*
B.04	50	100	121	121	120	0.83	0.83	0*	0.01
B.05	50	100	107	107	107	0.00	0.00	0*	0.01
B.06	50	100	121	121	121	0.00	0.00	0*	0*
B.07	75	94	204	204	204	0.00	0.00	0.01	0.03
B.08	75	94	204	204	204	0.00	0.00	0*	0.03
B.09	75	94	205	204	204	0.49	0.00	0.01	0.06
B.10	75	150	195	191	191	2.09	0.00	0*	0.01
B.11	75	150	184	183	183	0.55	0.00	0*	0.03
B.12	75	150	178	176	176	1.14	0.00	0*	0*
B.13	100	125	306	306	300	2.00	2.00	0*	0.11
B.14	100	125	278	278	278	0.00	0.00	0*	0.03
B.15	100	125	292	282	282	3.55	0.00	0.01	0.06
B.16	100	200	219	219	219	0.00	0.00	0*	0.03
B.17	100	200	199	198	198	0.51	0.00	0*	0.03
B.18	100	200	228	226	224	1.79	0.89	0*	0.03
Moyenne						**1.68**	**0.29**	**0.00**	**0.03**

Tableau 4.1: Performance de HSP et HSPI sur la classe B

* : Temps < 10^{-6} secondes

Inst.	n	m	Z_{SPH}	Z_{ISPH}	Z*	Gap HSP	Gap HSPI	T_{HSP}	T_{HSPI}
C.01	500	625	1394	1381	1379	1.09	0.15	0.82	9.85
C.02	500	625	1430	1418	1415	1.06	0.21	1.45	17.01
C.03	500	625	1407	1390	1385	1.59	0.36	1.39	31.56
C.04	500	625	1433	1415	1415	1.27	0.00	1.43	46.81
C.05	500	625	1382	1363	1358	1.77	0.37	0.75	14.87
C.06	500	1000	1152	1138	1138	1.23	0.00	0.09	1.54
C.07	500	1000	1222	1216	1213	0.74	0.25	0.09	2.42
C.08	500	1000	1161	1154	1154	0.61	0.00	0.09	1.98
C.09	500	1000	1160	1156	1156	0.35	0.00	0.07	1.87
C.10	500	1000	1164	1152	1152	1.04	0.00	0.09	2.42
C.11	500	2500	752	748	748	0.53	0.00	0.07	1.46
C.12	500	2500	789	784	784	0.64	0.00	0.06	0.84
C.13	500	2500	789	782	781	1.02	0.13	0.06	1.26
C.14	500	2500	773	767	767	0.78	0.00	0.06	1.03
C.15	500	2500	770	768	768	0.26	0.00	0.06	1.24
Moyenne						**0.93**	**0.10**	**0.44**	**9.08**

Tableau 4.2: Performance de HSP et HSPI sur la classe C

Inst.	n	m	Z_{HSP}	Z_{HSPI}	Z*	Gap HSP	Gap HSPI	T_{HSP}	T_{HSPI}
D.01	1000	1250	2703	2681	2678	0.93	0.11	10.73	211.27
D.02	1000	1250	2743	2717	2712	1.14	0.18	5.53	149.06
D.03	1000	1250	2724	2714	2714	0.37	0.00	5.32	50.63
D.04	1000	1250	2728	2708	2701	1.00	0.26	10.31	102.16
D.05	1000	1250	2748	2734	2727	0.77	0.26	25.87	91.81
D.06	1000	2000	2331	2313	2309	0.95	0.17	0.35	10.69
D.07	1000	2000	2343	2331	2329	0.60	0.09	0.36	7.50
D.08	1000	2000	2336	2314	2313	0.99	0.04	0.37	9.05
D.09	1000	2000	2348	2327	2327	0.90	0.00	0.35	13.38
D.10	1000	2000	2275	2254	2251	1.07	0.13	0.35	9.20
Moyenne						**0.87**	**0.12**	**5.96**	**65.47**

Tableau 4.3: Performance de HSP et HSPI sur la classe D

Dans les tableaux 4.4-4.5 ci-dessous, on présente la synthèse des résultats.

	Gap moyen de HSP	Gap moyen de HSPI
Classe B	1.68	0.29
Classe C	0.93	0.10
Classe D	0.87	0.12
Gap moyen total	**1.23**	**0.18**

Tableau 4.4: Gaps moyens de HSP et HSPI

	Temps moyen de HSP	Temps moyen de HSPI
Classe B	$< 10^{-6}$	0.03
Classe C	0.44	9.08
Classe D	5.96	65.47
Temps moyen	**2.13**	**24.86**

Tableau 4.5: Temps moyens en secondes de HSP et HSPI

En terme de comparaison des deux versions de l'heuristique, nos observations sont les suivantes:

- HSPI est plus performante que HSP. En effet, le Gap moyen de HSP par rapport à l'optimum est de 1.23% alors que celui de HSPI est de 0.18%.

- HSP est extrêmement rapide avec un temps moyen total de calcul de 2.13 secondes. Pour la classe B, HSP converge en un temps inférieur à 10^{-6} secondes.

- Il est clair que HSPI prend k fois plus de temps que HSP, où k est le nombre d'arbres couvants générés pour une instance donnée. Mais, HSPI reste rapide, car par exemple pour la classe B son temps moyen est de 0.03 secondes.

4.8 Conclusion

Pour avoir une borne supérieure du PCSTP, nous avons développé une heuristique *span-and-prune* basée sur une optimisation à deux phases. On part d'un arbre couvrant du graphe G (la phase *spanning*). Ensuite, on transforme l'arbre en un graphe acyclique, sur lequel le *pruning* revient à résoudre un problème du plus long chemin avec contrainte de capacité (PLCC).

Le PLCC est résolu de manière exacte en appliquant la relaxation lagrangienne, telle qu'elle était présentée par Minoux [159], suivie d'une procédure de classement [98]. Il s'agit d'une méthode approchée qui consiste à résoudre de manière exacte deux problèmes d'optimisation combinatoire.
Nous avons aussi démontré un nouveau résultat de complexité du PCSTP et nous avons présenté une étude de la performance du pire cas (*worst case)* de l'heuristique.

Deux variantes de l'heuristique (HSP et HSPI) ont été présentées, implémentées et testées sur 43 instances benchmark de *SteinLib* [131]. Les résultats sont extrêmement encourageants et seront utilisées pour appliquer une réduction du graphe afin de résoudre de manière exacte le PCSTP telle qu'elle est présentée dans le chapitre suivant.

Chapitre 5

Résolution exacte par la méthode de coupes et branchements

5.1 Introduction

Dans ce chapitre, nous nous intéressons à la résolution exacte du PC-STP. Ce dernier fait partie des problèmes de conception et d'optimisation des réseaux qui constituent une classe de problèmes d'optimisation combinatoire très difficile à résoudre par les techniques traditionnelles de la recherche opérationnelle.

Nous développons deux nouvelles formulations mathématiques exponentielles basées sur les coupes DCF et UDCF. De nouvelles inégalités valides sont aussi décrites. En plus, nous comparons les relaxations des différentes formulations (DCF et UDCF et MSTF).

En utilisant l'excellente borne supérieure de l'heuristique HSPI, une réduction du graphe (preprocessing) est aussi proposée sous deux approaches: un preprocessing basé sur la relaxation lagrangienne et un autre basé sur la relaxation linéaire. Une réduction selon les coûts et des tests de connexité du graphe sont aussi proposés.

Après avoir réduit le graphe, une procédure exacte de branch-and-cut est appliquée. Une étude expérimentale montre l'efficacité de la procédure proposée qui résoud à l'optimalité des instances Benchmark de grande taille atteignant 1000 noeuds.

5.2 Formulations mathématiques basées sur les coupes

Deux formulations basées sur les coupes sont proposées. Il s'agit de formulation avec un nombre exponentielle de contraintes.

5.2.1 Formulation non orientée UDCF

Partant du graphe connexe non orienté $G = (V, E)$, on prend comme vecteurs des variables de décision, les vecteurs $x \equiv (x_e)_{e \in E}$ et $y \equiv (y_j)_{j \in V^*}$ définis de la façon suivante:

$$\begin{aligned} x_e &= 1, \text{ si l'arête } e \in E \text{ appartient à la solution,} \\ &= 0, \text{ sinon,} \\ y_j &= 1, \text{ si le sommet } j \in V^* \text{ est couvert par la solution,} \\ &= 0, \text{ sinon.} \end{aligned}$$

Nous développons une formulation du problème non orientée et basée sur les coupes, dont la description est la suivante:

$$\textbf{UDCF}: \quad \text{Minimiser} \quad \sum_{e \in E} c_e x_e + \sum_{j \in V^*} \gamma_j (1 - y_j) \tag{63}$$

sous contraintes:

$$\sum_{j \in V^*} p_j y_j \geq Q, \tag{64}$$

$$\sum_{e \in \delta(S)} x_e \geq y_k, \qquad \forall S \subset V,\ 0 \in S,\ k \in V \setminus S, \tag{65}$$

$$x_e \in \{0, 1\}, \qquad \forall\ e \in E, \tag{66}$$

$$y_j \in \{0, 1\}, \qquad \forall\ j \in V^*. \tag{67}$$

On définit $\forall S \subset V,\ \delta(S) = \{e = (i, j) \in E : i \in S \text{ et } j \in V \backslash S\}$. C'est la coupe induite par S.

La contrainte (65) exprime le fait que si un noeud k, $k \in V^*$, est couvert par la solution ($y_k = 1$) alors nécessairement, il existe une chaine le relient au noeud racine 0. Cette contrainte (65) est aussi souvent noté par une contrainte des coupes et se caractérise par son nombre exponentiel.

On remarque qu'il y a d'autres contraintes valides qui sont automatiquement vérifiées par toute solution optimale du PCSTP telle que les contraintes qui suivent:

$$\sum_{e \in E} x_e = \sum_{j \in V^*} y_j \tag{68}$$

Cette contrainte (68) relie le nombre d'arête du sous-arbre solution au nombre de ses noeuds.

Les contrainte (69) et (70) traduisent le fait qu'une arête $e = \{i, j\} \in E$ n'est incluse dans la solution que si et seulement si les sommets i et j sont couverts par la solution.

$$x_e \leq y_i, \qquad \forall\ e = \{i, j\} \in E, \tag{69}$$

$$x_e \leq y_j, \qquad \forall\ e = \{i, j\} \in E, \tag{70}$$

Pour tout $k \in V^*$, si on prend comme sous-ensemble $S = V \backslash \{k\}$, la contrainte (65) devient dans ce cas précis:

$$\sum_{e \in \delta(k)} x_e \geq y_k, \qquad \forall k \in V^*, \tag{71}$$

où, $\delta(k) = \{(i, j) \in E : i \in S \text{ et } j = k\}$.

Notons ϑ_j le nombre minimal d'arêtes entre le noeud racine 0 et le noeud j, $j \in V^*$. L'ensemble des noeuds V^* sera ainsi partitionné en K sous-ensembles $W_1, W_2, ..., W_K$ telque $W_k = \{j \in V^* / \vartheta_j = k\}$. On cherche le plus grand entier k tel que $\sum_{j \in W_1 \cup W_2 \cup ... \cup W_P} p_j < Q$. Et on ajoute les $2P + 1$ contraintes suivantes:

$$\sum_{e \in \delta(W_k : W_{k+1})} x_e \geq 1 \quad \forall k = 1, ..., P \tag{72}$$

Où l'ensemble $\delta(W_k : W_{k+1}) = \{e = (i, j) \in E \text{ tels que } i \in W_k \text{ et } j \in W_{k+1}\}$.

Dans la suite, nous appelons EUDCF cette version renforcée et améliorée du modèle UDCF auquel on a ajouté les inégalités valides (68)-(72).

5.2.2 Formulation orientée DCF

Au lieu de formuler le PCSTP sur le graphe non orienté $G = (V, E)$, on considère une formulation *orientée* basée sur les coupes (Directed Cut

Formulation DCF) définie sur le graphe orienté $B = (V, A)$. Ce dernier est obtenu à partir du graphe initial G en remplaçant chaque arête $e = \{i, j\} \in E$ par deux arcs orientés (i, j) et (j, i) dont les coûts associés sont $c_{ij} = c_{ji} = c_e$. Ainsi, on peut affirmer que le PCSTP revient à chercher une arborescence optimale du graphe B dont la racine est le noeud 0.

Notons par z_{ij}, $(i, j) \in A$, la variable binaire qui prend la valeur 1 si l'arc (i, j) appartient à l'arborescence et 0 sinon. On définit aussi une variable binaire y_j, $j \in V^* = V \setminus \{0\}$, qui prend la valeur 1 si le noeud j appartient à l'arborescence et 0 sinon. En utilisant ces variables de décision, le PCSTP peut être formulé comme suit.

$$\textbf{DCF}: \quad \text{Minimiser} \quad \sum_{(i,j)\in A} c_{ij} z_{ij} + \sum_{j \in V^*} \gamma_j (1 - y_j) \tag{73}$$

sous contraintes:

$$\sum_{j \in V^*} p_j y_j \geq Q, \tag{74}$$

$$\sum_{(i,j)\in\delta^+(S)} z_{ij} \geq y_k, \qquad \forall S \subset V,\ 0 \in S,\ k \in V \setminus S, \tag{75}$$

$$z_{ij} \in \{0, 1\}, \qquad \forall\ (i, j) \in A, \tag{76}$$

$$y_j \in \{0, 1\}, \qquad \forall\ j \in V^*, \tag{77}$$

où, pour tout sous-ensemble non vide $S \subset V$, on définit $\delta^+(S) = \{(i, j) \in A : i \in S \text{ et } j \in V \backslash S\}$.

La fonction objectif (73) minimise la somme des coûts des arcs appartenant à l'arborescence et des pénalités des noeuds non couverts par cette solution. La contrainte (74) exige que le profit total collecté doit être au moins égal au quota spécifié. Les contraintes (76) et (77) indiquent que les variables de decision sont de type binaire.

La contrainte (75) est dite la contrainte de *connectivité* car elle exprime le fait que si un noeud k est couvert par la solution (i.e $y_k = 1$) alors il existe nécessairement un chemin qui le connecte au noeud racine 0. Fischetti [66] a montré que cette contrainte de connectivité est équivalente à la containte d'élimination de sous tours suivante:

$$\sum_{(i,j)\in E(S)} z_{ij} \geq \sum_{i \in S\setminus\{k\}} y_i \quad \forall k \in S,\ \forall S \subset V,\ |S| \geq 2 \tag{78}$$

Cette contrainte (78) est une généralisation de la contrainte d'élimination de sous-tours introduite par Dantzig et. al en 1954 [43] pour le problème de voyageur de commerce.

Nous indiquons que cette formulation (73)-(77) peut être considérée comme une géneralisation de la formulation *Directed Cut-Based Node Variable* (DCBNV) proposée à la base pour la variante *Node Weighted Steiner Tree Problem* (**NSP**) que nous avons décrite dans le paragraphe (2.8.1). Cette formulation DCBNV s'est révélée particulièrement intéressante depuis que Chopra et Tsai [31] ont démontré que sa relaxation linéaire produit d'excellentes bornes inférieures pour le problème de l'arbre de Steiner.

Renforcement de la relaxation linéaire du Modèle DCF

Il est assez clair que la performance de la relaxation linéaire est d'une importance cruciale pour la solution effective du programme en nombre entier. Dans ce paragraphe, nous proposons de nouvelles classes d'inégalités valides pour renforcer la relaxation linéaire du modéle DCF. Il s'agit de contraintes valides supplémentaires qui sont automatiquement vérifiées par toute solution optimale du problème.

Dans toute la suite, V et A représentent l'ensemble de noeuds et des arcs obtenus après l'application d'un preprocessing réduisant la taille des instances, tel que nous allons décrire au paragraphe 6.3.

Inégalités simples Manifestement, pour une solution réalisable binaire, l'inégalité suivante est vérifiée $z_{ij} + z_{ji} \leq 1$, $\forall\ (i,j) \in A$.
En plus, $\forall\ i \in V^*$, $y_i = 0 \Rightarrow\ \ z_{ij} = z_{ji} = 0$, $\forall\ (i,j) \in A$. On peut combiner ces contraintes de manière linéaire comme suit:

$$z_{ij} + z_{ji} \leq y_j, \qquad \forall\ j \in V^*,\ (i,j) \in A. \tag{79}$$

Cette contrainte (79) traduit le fait qu'il suffit que l'un des arcs (i,j) ou (j,i) soit pris dans l'arborescence optimale pour que les sommets i et j soient nécéssairement couverts par cette arborescence.

De plus, comme les poids des arcs sont de signe positif, il existe une arborescence optimale où chaque noeud couvert $j \in V^*$ a exactement un unique arc incident (un seul arc entrant). Ainsi, on ajoute la contrainte qui

suit.

$$\sum_{i \in \delta^-(j)} z_{ij} = y_j, \qquad \forall j \in V^*, \tag{80}$$

où, $\delta^-(j)$ est l'ensemble des prédécesseurs du sommet $j \in V^*$.

Inégalités basées sur le flot L'inégalité (81) , qui suit, a déjà été proposée pour le problème de l'arbre de Steiner [171]. Elle exige qu'un noeud $j \in V^*$ n'admette aucun arc sortant, s'il n'admet pas d'arcs entrants. Il s'agit de la contrainte suivante:

$$\sum_{i \in \delta^-(j)} z_{ij} \geq z_{jk}, \qquad \forall j \in V^*,\ (j,k) \in A. \tag{81}$$

On note que cette contrainte (81) est différente de la contrainte (82) valide pour le problème de Steiner et qui s'écrit comme suit

$$\sum_{k \in \delta^-(i)} z_{ki} \leq \sum_{k \in \delta^+(i)} z_{ik}, \quad \forall i \in S, \tag{81'}$$

où, $\delta^+(i) = \{j \in V : (i,j) \in A\}$ est l'ensemble des successeurs du sommet i et S est l'ensemble des noeuds de Steiner. Il s'agit de la "*flow balance constraint*" introduite par Koch et martin en 1998 [130] pour le problème de steiner classique. Elle exprime le fait que pour chaque noeud de steiner, le flot total sortant doit être supérieur au flot total entrant.

Inégalités de recouvrement Pour chaque noeud $j \in V^*$, on note ϑ_j le nombre minimal d'arcs du chemin orienté P entre le noeud racine 0 et le noeud j. En plus, pour chaque entier h, on définit le sous-ensemble de noeuds $S_h = \{j \in V : \vartheta_j = h\}$.
Soit h^* le plus petit entier satisfaisant l'inégalité suivante :

$$\sum_{h=1}^{h^*} \sum_{j \in S_h} p_j \geq Q.$$

Il est clair que pour toute solution réalisable, il existe forcément un chemin qui contient au moins un noeud de chacun des sous-ensembles successifs $V_1, ...V_{h^*}$. Ainsi, on obtient la contrainte qui suit.

$$\sum_{(i,j) \in \delta(S_{h-1}, S_h)} z_{ij} \geq 1, \qquad \forall\ h = 1, ..., h^*, \tag{82}$$

où $\delta(S_{h-1}, S_h)) \equiv \{(i,j) \in A : i \in S_{h-1} \text{ et } j \in S_h\}$.

Dans la suite, nous appelons EDCF cette version renforcée et améliorée du modèle DCF, auquel nous avons ajouté les inégalités valides (79)-(82).

5.3 Réduction du graphe

L'objectif de la réduction du graphe ou du preprocessing est de simplifier le graphe du problème par l'élimination des arêtes et des sommets qui n'ont pas la possibilité d'appartenir à la solution optimale. Le but d'une telle opération est de réduire la taille des instances et par conséquent d'accélérer leur résolution.

Dans cette partie, nous décrivons successivement deux stratégies de réduction: un preprocessing basé sur la relaxation lagrangienne et un autre basée sur la relaxation linéaire. Une réduction basée sur les coûts ainsi que des tests de connexité sont aussi détaillés.

5.3.1 Preprocessing basé sur la relaxation lagrangienne

Dans le chapitre 3, nous avons vérifié qu'appliquer la variante SGA6 de l'algorithme du sous-gradient pour la résolution du problème dual de la décomposition lagrangienne (Relaxation 2) donne souvent les meilleures bornes inférieures pour le PCSTP. La solution primale obtenue par SGA6 n'est pas nécessairement réalisable, mais nous allons l'exploiter pour réduire la taille des instances.

Nous commençons par rappeler la formulation MSTF que nous avons introduite pour le GPCSTP au paragraphe 4.2:

$$\textbf{MSTF:} \quad \text{Minimiser} \sum_{\{i,j\} \in \bar{E}} c_{\{i,j\}} x_{\{i,j\}} \tag{83}$$

sous contraintes:

$$\sum_{j \in V^*} p_j x_{\{s,j\}} \leq \bar{Q}, \tag{84}$$

$$x_{\{s,j\}} + x_{\{i,j\}} \leq 1, \qquad \forall j \in V^*,\ \{i,j\} \in E, \tag{85}$$

$$x_{\{s,1\}} = 1, \tag{86}$$

$$\sum_{\{i,j\}\in\bar{E}} x_{\{i,j\}} = \left|\bar{V}\right| - 1, \tag{87}$$

$$\sum_{\{i,j\}\in\bar{E}:i,j\in S} x_{\{i,j\}} \leq |S| - 1, \qquad \forall S \subset \bar{V},\ 3 \leq |S| \leq \left|\bar{V}\right| - 2, \tag{88}$$

$$x_{\{i,j\}} \in \{0,1\}, \qquad \forall \{i,j\} \in \bar{E}, \tag{89}$$

où, la variable binaire $x_{\{i,j\}}$ est égale à 1 si l'arête $\{i,j\}$ appartient à la solution optimale, 0 sinon. Pour appliquer la Relaxation 2 (paragraphe 4.3.3), nous définissons pour chaque variable $x_{\{s,j\}}$, $j \in V^*$, deux copies notées respectivement u_j et v_j. Nous dupliquons aussi chaque variable $x_{\{i,j\}}$, $\{i,j\} \in E$, en notant sa copie $w_{\{i,j\}}$. Ainsi, on obtient la formulation qui suit.

$$\text{Minimiser} \quad \sum_{\{i,j\}\in\bar{E}} c_{\{i,j\}} x_{\{i,j\}} \tag{90}$$

sous contraintes:

$$\sum_{j\in V^*} p_j u_j \leq \bar{Q}, \tag{91}$$

$$u_j \in \{0,1\}, \qquad \forall j \in V^*, \tag{92}$$

$$v_j + w_{\{i,j\}} \leq 1, \qquad \forall j \in V^*,\ \{i,j\} \in E, \tag{93}$$

$$v_j,\ w_{\{i,j\}} \in \{0,1\}, \qquad \forall\ j \in V^*,\ \{i,j\} \in E, \tag{94}$$

$$x_{\{s,0\}} = 1, \tag{95}$$

$$\sum_{\{i,j\}\in\bar{E}} x_{\{i,j\}} = \left|\bar{V}\right| - 1, \tag{96}$$

$$\sum_{\{i,j\}\in\bar{E}:i,j\in S} x_{\{i,j\}} \leq |S| - 1, \qquad \forall S \subset \bar{V},\ 3 \leq |S| \leq \left|\bar{V}\right| - 2, \tag{97}$$

$$x_{\{i,j\}} \in \{0,1\}, \qquad \forall \{i,j\} \in \bar{E}, \tag{98}$$

$$x_{\{s,j\}} - u_j = 0, \qquad \forall j \in V^*, \tag{99}$$

$$x_{\{s,j\}} - v_j = 0, \qquad \forall j \in V^*, \tag{100}$$

$$x_{\{i,j\}} - w_{\{i,j\}} = 0, \qquad \forall \{i,j\} \in E. \tag{101}$$

Notons que $x_{\{s,j\}} = 1$, ou $u_j = 1$, ou $v_j = 1$ signifie que le noeud $j \in V^*$ n'est pas couvert par la solution du PCSTP. En plus, $x_{\{i,j\}} = 1$ ou $w_{\{i,j\}} = 1$ signifie que l'arête $\{i,j\} \in E$ appartient à la solution du PCSTP.

Les multiplcateurs de lagrange associés aux contraintes (99), (100) et (101) sont respectivement notés α_j, $j \in V^*$, β_j, $j \in V^*$ et $\delta_{\{i,j\}}$, $\{i,j\} \in E$. La fonction lagrangienne correspondante s'écrit

$$\theta(\alpha,\beta,\delta) = \text{Min} \sum_{\{i,j\}\in\bar{E}} \tilde{c}_{\{i,j\}} x_{\{i,j\}} - \sum_{j\in V^*} \alpha_j u_j - (\sum_{j\in V^*} \beta_j v_j + \sum_{\{i,j\}\in E} \delta_{\{i,j\}} w_{\{i,j\}}) \tag{102}$$

sous contraintes: $(91) - (98)$,
où,

$$\tilde{c}_{\{i,j\}} = c_{\{i,j\}} + \delta_{\{i,j\}}, \quad \forall \{i,j\} \in E, \tag{103}$$
$$\tilde{c}_{\{s,j\}} = \alpha_j + \beta_j + \gamma_j, \quad \forall\, j \in V^*. \tag{104}$$

Nous avons déjà détaillé que le calcul de $\theta(\alpha,\beta,\delta)$ revient à résoudre les trois problèmes suivant:

- Un problème de sac à dos (KP) défini par (91) et (92),
- Un problème d'ensemble stable de poids maximal (MWSP) défini par (93) et (94),
- Un problème d'arbre couvrant de poids minimal (MST) défini par (95)-(98).

En conséquence, on résoud le problème dual lagrangien défini par

$$\textbf{LD} : \text{Maximiser } \theta(\alpha,\beta,\delta), \tag{105}$$

afin de dériver une borne inférieure pour le PCSTP. Nous avons conclu que les meilleurs résultats sont obtenus en résolvant LD de manière approchée par l'algorithme du sous-gradient dévié SGA6 (la stratégie *ADS* [199]).

Appelons $(\alpha^*,\beta^*,\delta^*)$ le vecteur des meilleurs multiplicateurs de lagrange obtenu à la terminaison de l'algorithme SGA6 et (x^*,u^*,v^*,w^*) la solution primale correspondante qui détermine $\theta(\alpha^*,\beta^*,\delta^*)$ selon (102). Ainsi, x^* est le vecteur d'incidence d'un arbre couvrant $\mathcal{T}$ de $\bar{G}$, u^* est le vecteur d'incidence d'une solution de type sac à dos réalisable, et (v^*,w^*) est le vecteur d'incidence d'un ensemble stable pour le MWSP. Notons par T^*, K^*

et S^* les valeurs objectives correspondantes respectives. L'égalité suivante est vérifié:

$$\theta(\alpha^*, \beta^*, \delta^*) = T^* - K^* - S^*. \tag{106}$$

Pour chaque arête $e = \{i, j\} \in E$, appelons K_e la valeur de la solution optimale du problème de type sac à dos:

$$\text{Maximiser} \quad \sum_{k \in V \setminus \{0,i,j\}} \alpha_k^* u_k \tag{107}$$

sous contraintes:

$$\sum_{k \in V \setminus \{0,i,j\}} p_k u_k \leq \bar{Q}, \tag{108}$$

$$u_k \in \{0, 1\}, \qquad \forall\, k \in V \setminus \{0, i, j\}. \tag{110}$$

Il est évident que si $u_i^* = u_j^* = 0$, alors $K_e = K^*$.

Ensuite, considérons le sous-problème MST. Supposons que l'arête $e \in E$ n'est pas couverte par $\mathcal{T}$, alors, ajouter cette arête e à l'arbre $\mathcal{T}$ va créer exactement un cycle $\mathcal{C}_e$. Appelons $f \in \mathcal{C}_e \setminus \{e\}$ l'arête de $\mathcal{C}_e$ ayant le coût réduit le plus élevé (i.e., $f = \arg \max_{\{i,j\} \in \mathcal{C}_e \setminus \{e\}} \tilde{c}^*_{\{i,j\}}$ avec $\tilde{c}^*_{\{i,j\}} = c_{\{i,j\}} + \delta^*_{\{i,j\}}, \forall \{i, j\} \in E$).
A présent, pour chaque arête $e = \{i, j\} \in E$, nous définissons

- $T_e = T^*$ si $x_e^* = 1$, sinon, $T_e = T^* + \tilde{c}_e^* - \tilde{c}_f^*$,
- $S_e = S^*$ si $w_e^* = 1$, sinon $S_e = S^* + \bar{\delta}_e$, où $\bar{\delta}_e$ est le coût réduit de l'arête e qui est obtenue après la résolution du MWSP en appliquant la programmation linéaire, afin d'évaluer $\theta(\alpha^*, \beta^*, \delta^*)$.

Notons par UB la borne supérieure obtenue par HSPI. Considérons le résultat suivant:

Proposition 21 *une arête $e \in E$ ne peut pas appartenir à la solution optimale si*

$$T_e - K_e - S_e > UB. \tag{111}$$

Preuve. Il suffit d'observer que si l'arête $e = \{i, j\}$ est contrainte d'appartenir à la solution optimale, alors il faut ajouter les contraintes $x_e = 1$, $u_i = u_j = 0$ et $w_e = 1$ aux contraintes (90)-(101).
En examinant la valeur du dual lagrangien de la même solution duale $(\alpha^*, \beta^*, \delta^*)$, et en résolvant les sous-problèmes correspondants KP, MST, et en dérivant une borne supérieure du sous-problèmes correspondant MWSP, on trouve qu'une borne inférieure valide du PCSTP, contenant l'arête e, est $T_e - K_e - S_e$. On conclut que le résultat est immédiat. ■

Ce résultat représente un moyen pratique pour éliminer les arêtes non nécéssaires du graphe avant de résoudre le PCSTP. Après l'application de ce preprocessing, si on constate qu'on a éliminé toutes les arêtes adjacentes à un noeud i, alors ce noeud i est aussitôt éliminé.

5.3.2 Preprocessing basé sur la relaxation linéaire

Une deuxième technique de réduction du graphe se base sur le principe de fixer à priori les valeurs de quelques variables du problème. Le principe de cette stratégie est le suivant:
Considérons le programme d'optimisation en nombres entiers suivant:

$$\textbf{IP} : \quad \min\{cx : Ax \leq b,\ x \in \{0,1\}^n\},$$

Notons z_H la valeur de la solution heuristique du problème IP.

Proposition 22 *[162] Soit $\bar{x}$ une solution optimale de la relaxation linéaire LP,*

$$\textbf{LP} : \quad \min\{cx : Ax \leq b,\ x \in [0,1]^n\},$$

Notons z_{LP} la valeur objective optimale et $\bar{c}_j$ le coût réduit associé à la variable $\bar{x}_j$ $(\forall j = 1, ..., n)$.

Si $\bar{x}_j = 0$ et $z_{LP} + \bar{c}_j > z_H$, alors $x_j = 0$ pour toute solution optimale de IP.

Si $\bar{x}_j = 1$ et $z_{LP} - \bar{c}_j > z_H$, alors $x_j = 1$ pour toute solution optimale de IP. ■

Par conséquent, considérons la relaxation linéaire du modèle EDCF. Supposons que la relaxation linéaire donne comme solution optimale $(\bar{y}, \bar{z})$ et

comme valeur objective LR (les détails de la procédure de résolution sont exposés dans le paragraphe 6.4).

Commençons par noter $\bar{c}_{ij}$ et $\bar{\gamma}_j$ les coûts réduits respectifs aux variables z_{ij}, $(i,j) \in A$ et y_j, $j \in V^*$. Nous proposons les tests de réduction suivants:

(a) un arc $(i,j) \in A$ ne peut pas appartenir à une solution optimale si

$$\bar{z}_{ij} = 0 \text{ et } LR + \bar{c}_{ij} > UB; \tag{112}$$

(b) un noeud $j \in V^*$ ne peut pas appartenir à une solution optimale si

$$\bar{y}_j = 0 \text{ et } LR + \bar{\gamma}_j > UB; \tag{113}$$

(c) un arc $(i,j) \in A$ appartient nécessairement à une solution optimale si

$$\bar{z}_{ij} = 1 \text{ et } LR - \bar{c}_{ij} > UB; \tag{114}$$

(d) un noeud $j \in V^*$ appartient nécessairement à une solution optimale si

$$\bar{y}_j = 1 \text{ et } LR - \bar{\gamma}_j > UB. \tag{115}$$

Dans le cas où un arc $(i,j) \in A$ appartient nécessairement à une solution optimale, nous fusionnons les noeuds i et j en un seul noeud v en satisfaisant $p_v = p_i + p_j$ et $\gamma_v = \gamma_i + \gamma_j$, et en fixant $y_v = 1$. En plus, chaque arc $(k,i) \in A$ est remplacé par un arc (k,v) ayant le même coût. Chaque arc sortant des noeuds i ou j est remplacé par un arc sortant du noeud v ayant le même coût. Il est évident que si après cette transformation, il y a deux arcs sortant du noeud v au même noeud k, alors on élimine celui qui a le coût le plus élévé.

5.3.3 Preprocessing basé sur les coûts

Nous avons aussi dévéloppé un *preprocessing basé sur les coûts* à travers les régles qui suivent.
Il est possible de fixer d'autres variables en observant que si un certain arc

$(i,j) \in A$ vérifie l'inégalité $c_{ij} \leq \gamma_j$, alors la condition suivante est vérifiée à l'optimalité: si $y_i = 1 \Rightarrow y_j = 1$. Dans ce cas, on peut imposer l'inégalité $y_i \leq y_j$.
Par conséquent, si pour un noeud $j \in V^*$, l'inégalité $c_{0j} \leq \gamma_j$ est vérifiée, alors on prend $y_j = 1$.
En plus, comme appliqué par Duin et Volgenant [52], si pour un arc $(i,j) \in A$ on a $c_{ij} \leq \min\{\gamma_i, \gamma_j\}$, alors l'égalité $y_i = y_j$ est vérifié.

5.3.4 Tests de connexité

Une fois un type de preprocessing appliqué, nous testons si le graphe réduit obtenu est connexe en appliquant la procédure suivante:

On considére le graphe initial G. On associe à chaque arête éliminée par le preprocessing un coût unitaire et aux arêtes restantes des coûts nuls. On résoud le problème de l'arbre couvrant de poids minimal (MST) avec ces nouveaux coûts. Si le poids de l'arbre optimale est strictement positif alors le graphe est non connexe, sinon le graphe réduit est connexe. En effet, ceci s'explique par le fait que les arêtes restantes ne peuvent pas seules former un arbre (connexe), on a besoin d'autres arêtes qui sont déjà éliminées par le preprocessing.
Dans le cas où le graphe réduit est non connexe, on élimine toutes les composantes connexes qui ne contiennent pas le noeud racine 0. Et, on vérifie la condition de réalisabilité suivante: il faut que la somme des profits des noeuds de la composante connexe qui contient le noeud racine 0 doit être supérieure ou égale au quota.

Dans notre implementation, les tests de preprocessing décrits ci-dessus sont appliqués comme suit. D'abord, on fait appel à la Proposition 30 pour réduire le graphe. Ensuite, on applique le preprocessing basé sur la relaxation linéaire ainsi que les tests basés sur les coûts afin d'éliminer des arcs et des noeuds supplémentaires. En plus, après la réalisation de chaque reduction du graphe, tous les noeuds isolés sont éliminés ainsi que tous les noeuds n'ayant pas d'arcs incidents. Par suite, on vérifie la connexité du graphe réduit. En effet, l'élimination des arêtes et des arcs durant la phase du preprocessing pourrait produire un graphe reduit non connexe. Enfin, nous avons dévéloppé une procédure de *propagation.* En effet, chaque fois que la variable d'un noeud est fixée à une nouvelle valeur, nous propageons cette décision aux noeuds adjacents.

Après avoir appliqué cette stratégie de réduction du graphe, nous appliquons une procédure exacte de résolution qui se base sur la méthode de génération de coupes telle que nous la présentons dans le paragraphe qui suit.

5.4 Procédure exacte de type coupes et branchements

Nous décrivons une méthode exacte pour résoudre le modèle EDCF qui a été présentée dans le paragraphe 6.2.2. Comme ce modèle contient un nombre exponentiel de contraintes, nous proposons de le résoudre en appliquant une procédure de plans coupants. En effet, les contraintes des coupes (75) ne sont pas *a priori* toutes considérées et seules les contraintes violées sont itérativement générées et ajoutées au modèle.

Plus précisément, on applique d'abord le preprocessing basé sur la relaxation lagrangienne en utilisant la solution heuristique générée par l'heuristique HSPI, afin de réduire le graphe initial G. Ensuite, on construit un modèle initial EDCF qui inclut toutes les contraintes à l'exception des contraintes (75). Par suite, nous résolvons la relaxation linéaire du modèle en utilisant un solveur, tel que CPLEX, pour obtenir une solution optimal $(\bar{y}, \bar{z})$. Dans le but de vérifier s'il existe éventuellement une ou plusieurs contrainte de coupe (75) violée, nous devons résoudre un problème de séparation pour chaque $k \in V^*$ telque $\bar{y}_k > 0$, défini comme suit.

$$\textit{Déterminer } W_k \subset V \textit{ satisfaisant } 0 \in W_k \textit{ et } k \in V \setminus W_k, \textit{ telque } \sum_{(i,j)\in\delta^+(W_k)} \bar{z}_{ij} < \bar{y}_k, \quad \textit{ou vérifier qu'un tel sous-ensemble n'existe pas.} \tag{116}$$

Ce problème revient à chercher une coupe minimale qui sépare le noeud racine 0 du neoud k dans le graphe orienté B du paragraphe 6.2.2, où la capacité de chaque arc $(i, j) \in A$ est $\bar{z}_{ij}$.
Notons par ϕ_k la valeur du flot maximal circulant de la source 0 vers le puit k avec ces capacitiés sur les arcs. En appliquant le théormèe flot-max-coupe-min de Ahuja et al. [2], nous obtenons la condition: si $\phi_k < \bar{y}_k$, alors le sous-ensemble de noeuds correspondant à la coupe minimale résolve (116) produit une coupe violée (75).
Ainsi, si une ou plusieurs coupes violées sont déterminées, alors les contraintes

correspondantes sont ajoutées au modèle, et le processus est reitéré jusqu'à trouver une solution réalisable.

A la suite, le graphe est réduit en appliquant les strétégies du preprocessing basé sur la relaxation linéaire et sur les coûts discutées dans les paragraphes 6.3.2 et 6.3.3. Enfin, en utilisant le graphe réduit resultant, on continue la stratégie de génération de coupes discutée ci-dessus, sauf qu'à présent, chaque modèle relaxé est résolu comme un programme en nombre entier.

Une synthèse de l'approche proposée est présentée ci-dessous.

Etape 0: Saisir une instance du PCSTP définie sur un graphe non orienté G.

Etape 1: Résoudre le dual lagrangien défini par (105) afin de dériver une borne inférieure LB_{LD} et calculer une borne supérieure UB en appliquant l'heuristique HSPI.

Etape 2: Test de termination 1: si $UB = LB_{LD}$, alors aller à Etape 7.

Etape 3: Appliquer le test du preprocessing basé sur la relaxation lagrangienne (voir proposition 30) pour réduire le graphe G et construire le graphe orienté associé B. Appliquer l'heuristique HSPI sur le graphe B. Si une meilleure solution est découverte, alors mettre à jour UB.
Test de termination 2: si $UB = LB_{LD}$, alors aller à Etape 7.

Etape 4: Résoudre la relaxation linéaire du Modèle EDCF en appliquant la stratégie de génération de coupes précitée.
Notons LB_{LP} la valeur de la solution optimale.
Test de termination 3: si $UB = LB_{LP}$, alors aller à Etape 7.

Etape 5: Appliquer la stratégie du preprocessing basé sur la relaxation linéaire et sur les coûts pour réduire le graphe B.

Etape 6: Utiliser un solveur (CPLEX) pour résoudre le modèle EDCF qui inclut les contraintes de coupes (75) qui étaient générées durant l'Etape 4, ensuite, générer toute contrainte (75) violée par la solution optimale binaire et reitérer Etape 6.
Si aucune contrainte violée n'est trouvée, alors aller à Etape 7.

Etape 7: Sortir la solution courante comme solution optimale.

Il est assez intéréssant de signaler que si une instance admet des coûts et des pénalités entiers, alors il est possible de remplacer LB par $\lceil LB \rceil$.

5.5 Etude expérimentale

5.5.1 Performance de la procédure exacte

Pour évaluer la performance de la procédure que nous proposons, nous l'avons implémentée sur un ordinateur Pentium IV 3.2 GHz avec 3.0 GB de RAM, en utilisant Microsoft Visual C++ comme language de programmation et CPLEX comme solveur les différents programmes linéaires et en nombre entier . Nous avons fixé le temps CPU maximal d'exécution à 3600 secondes. Comme pour l'heuristique *span-and-prune*, nous avons testé la procédure exacte sur les trois classes d'instances benchmark du *SteinLib* [131]. Les détails des instances sont disponibles au paragraphe 4.7.

Les résultats de la procédure exacte de type branch-and-cut sont présentés dans les tableaux 5.1-5.3. La signification des colonnes est la suivante:

- $Inst. \equiv$ le nom de l'instance,
- $n \equiv$ le nombre de noeuds,
- $m \equiv$ le nombre des arêtes,
- $Z^* \equiv$ La valeur de la solution optimale,
- $Temps \equiv$ le temps CPU total en secondes.

Inst.	n	m	Z*	Temps
B.01	50	63	140	0.21
B.02	50	63	153	0.24
B.03	50	63	128	0.21
B.04	50	100	120	0.42
B.05	50	100	107	0.26
B.06	50	100	121	0.43
B.07	75	94	204	0.82
B.08	75	94	204	0.32
B.09	75	94	204	0.58
B.10	75	150	191	0.54
B.11	75	150	183	0.25
B.12	75	150	176	0.32
B.13	100	125	300	0.95
B.14	100	125	278	0.73
B.15	100	125	282	0.82
B.16	100	200	219	0.51
B.17	100	200	198	0.45
B.18	100	200	224	1.28
Moyenne				**0.52**

Tableau 5.1: Performance de la procédure exacte sur la classe B

Inst.	n	M	Z*	Temps
C.01	500	625	1379	166.67
C.02	500	625	1415	51.57
C.03	500	625	1385	520.57
C.04	500	625	1415	71.80
C.05	500	625	1358	40.86
C.06	500	1000	1138	12.04
C.07	500	1000	1213	16.95
C.08	500	1000	1154	20.86
C.09	500	1000	1156	15.48
C.10	500	1000	1152	17.31
C.11	500	2500	748	20.57
C.12	500	2500	784	17.21
C.13	500	2500	781	28.79
C.14	500	2500	767	12.17
C.15	500	2500	768	20.70
Moyenne				**68.91**

Tableau 5.2: Performance de la procédure exacte sur la classe C

Inst.	n	m	Z*	Temps
D.01	1000	1250	2678*	---**
D.02	1000	1250	2712	1517.09
D.03	1000	1250	2714	267.56
D.04	1000	1250	2701	228.37
D.05	1000	1250	2727	281.82
D.06	1000	2000	2309*	---**
D.07	1000	2000	2329	99.38
D.08	1000	2000	2313	72.81
D.09	1000	2000	2327	95.13
D.10	1000	2000	2251	91.31
Moyenne				**331.69**

Tableau 5.3: Performance de la procédure exacte sur la classe D

(* la meilleure borne inférieure trouvée- ** Temps de résolution > 3600 secondes)

Inst.	*n*	*m*	*PDA*	*PDN*	*PFN*
B.01	50	63	29.37	20.00	26.00
B.04	50	100	40.00	24.00	32.00
B.09	75	94	43.09	22.67	26.67
B.13	100	125	7.60	5.00	9.00
B.18	100	200	25.50	11.00	20.00
C.01	500	625	33.68	24.00	20.60
C.02	500	625	20.48	14.60	22.20
C.03	500	625	7.04	4.00	11.20
C.05	500	625	2.48	2.40	13.80
C.07	500	1000	23.75	6.80	27.20
C.13	500	2500	70.28	4.40	83.00
D.01	1000	1250	13.12	11.20	26.90
D.02	1000	1250	6.08	5.40	16.50
D.04	1000	1250	2.04	1.40	14.50
D.05	1000	1250	2.36	1.80	14.90
D.06	1000	2000	30.88	10.30	32.00
Moyenne			**22.36**	**10.56**	**24.78**

Tableau 5.4: Performance des procédures de réduction pour les instances ayant *LB < UB*

A partir des tablaux 5.1-5.3., nous observons que l'approche proposée montre une excellente performance. En effet, elle résoud à l'optimum des instances du PCSTP de grande taille (1000 noeuds) avec des temps de calcul assez raisonables. Parmi les 43 instances tests, la solution optimale a été obtenue pour 41 instances. Pour les deux instances non résolues à l'optimalité, les pourcentages des gaps de la solution heuristique par rapport à l'optimum sont respectivement de 0.11% et 0.17%.

Dans le tableau 5.4, nous rapportons la performance du preprocessing pour les instances particulières vérifiant $UB > LB \equiv \max\{LB_{LR}, LB_{LP}\}$. Notons que pour cet ensemble d'instances, qui compte 16 instances parmi les 43 instances tests, toutes les strétgies de preprocessing décrite ci-dessus sont appliquées. La signification des colonnes est la suivante:

- $PDA \equiv$ le pourcentage des arcs éliminés,
- $PDN \equiv$ le pourcentage des noeuds éliminés,
- $PFN \equiv$ le pourcentage des noeuds dont les variables sont fixées à 1.

On ne rapporte pas le pourcentage des arcs dont les variables sont fixées à 1 parce que ce pourcentage est égal à zéro pour toutes les instances, sauf pour (C13) dont les variables de deux arcs sont fixées à 1.

A partir du tableau 5.4, nous observons que très souvent la taille des problèmes est significativement réduite malgrè le fait que les instances du *steinLib* sont déjà très peu dense. Cependant, l'efficacité des procédures de reduction est elle même dûe à la qualité de la borne supérieure implémentée. En effet, nous avons trouvé que la version itérative de l'heuristique *span-and-prune* HSPI donne d'excellentes bornes produisant des gaps par rapport à l'optimum pour les classes d'instances B, C et D respectivement de 0.29%, 0.10% et 0.12%.

Impact des inégalités valides

Afin d'examiner l'impact des contraintes valides (79)-(82), nous les avons enlevées de la formulation et nous avons exécuté la procédure exacte proposée pour les instances pour lesquelles $UB > LB$ à la racine. Les résultats sont présentés dans le Tableau 5.5. Dans ce tableau, la colonne "Ratio Temps" signifie le rapport du temps CPU obtenu avec la formulation simplifiée par rapport à celui de la formulation renforcée. A partir de ce tableau, nous observons que pour la majorité des instances résolues, le temps CPU augmente

de façon spectaculaire si les inégalités valides proposées ne sont pas prises en considération. En plus, quatre instances demeurent non résolues après avoir atteind la limite du temps maximal (3600 secondes).

Impact du preprocessing

D'abord, notons que cet ordre d'application des procédures de preprocessing (Etape 3 et 4) a été décidé suite à une étude empirique exhaustive qui a révélée que la meilleure performance est due à ce choix.

Afin d'évaluer l'efficacité des stratégies de réduction du graphe proposées, nous avons exécuté une version simplifiée de l'approche exacte qui inclue les contraintes valides (79)-(82), et où les routines du preprocessing sont sautées. Les résultats obtenus pour les instances de la classe D sont disponibles dans le tableau 5.6.

En dépit du fait que deux instances ont été résolues en moins de temps CPU, nous observons que la performance algorithmique globale s'est significativement détériorée et que seulement quatre instances ont été résolues à l'optimalité. D'où, l'intérêt du développement des stratégies de réduction des graphes.

Nous attirons l'attention du lecteur sur l'excellente performance de la formulation EDCF. En effet, nous avons aussi testé cette même procédure exacte avec la version non orientée de la formulation basée sur les coupes EU-DCF. Les résultats numériques sont extrêmement décevants, à savoir auncune instance des classes C et D n'a été résolue à l'optimum.

Inst.	*n*	*m*	*Temps*	*Ratio Temps*
B.01	50	63	0.28	1.289
B.04	50	100	0.47	1.111
B.09	75	94	0.83	1.431
B.13	100	125	4.61	4.824
B.18	100	200	2.78	2.168
C.01	500	625	172.98	1.038
C.02	500	625	75.01	1.454
C.03	500	625	521.51	1.002
C.05	500	625	80.38	1.97
C.07	500	1000	390.23	23.02
C.13	500	2500	81.35	2.82
D.01	1000	1250	---	---
D.02	1000	1250	---	---
D.04	1000	1250	---	---
D.05	1000	1250	435.61	1.55
D.06	1000	2000	---	---
Moyenne				**3.64**

Tableau 5.5: Impact des contraintes valides

Inst.	*n*	*m*	*Temps*	*Ratio Temps*
D.01	1000	1250	---	---
D.02	1000	1250	---	---
D.03	1000	1250	---	---
D.04	1000	1250	---	---
D.05	1000	1250	120.15	0.43
D.06	1000	2000	---	---
D.07	1000	2000	791.14	7.96
D.08	1000	2000	6.49	0.09
D.09	1000	2000	120.11	1.26
D.10	1000	2000	---	---

Tableau 5.6: Impact des procédures de réduction

Dans le paragraphe suivant, nous comparons empiriquement les relaxations linéaires des deux formulations basées sur les coupes EUDCF et EDCF ainsi que la relaxation lagrangienne 2 de la formulation MSTF.

5.5.2 Comparaison empirique des relaxations

Nous nous intéréssons aux relaxations linaires respectives des formulations EDCF et EUDCF. Nous avons tenté de les comparer théoriquement. Et aussi, nous avons cherché à comparer les relaxations linéaires des formulations basées sur les coupes par rapport à la relaxation lagrangienne la plus performante (Relaxation2) de la formulation basées sur les arbres FMST.

Notons par z_{LR-UD} et z_{LR-D} les solutions exactes respectives des problème de la relaxation linéaire de la formulation EUDCF et EDCF.

Considérons la formulation FMST. Suite à l'application de la décomposition lagrangienne (Relaxation 2) à FMST, le dual lagrangien est résolu de manière exacte en appliquant l'algorithme de génération de coupes stabilisé (paragraphe) et cette borne est notée par z_{LGR}.

On trouve que les trois différentes relaxations sont *incomparables deux à deux* à travers les instances du PCSTP suivantes:

Les données du problème (**P1**) sont les suivantes: ($n = 3$, $m = 6$, $Q = 29$)

arête $e \in E$	c_e
{0,1}	43
{0,2}	71
{0,3}	45
{1,2}	47
{1,3}	76
{2,3}	79

sommet $j \in V^*$	profit p_j	pénalité γ_j
1	20	10
2	13	6
3	25	12

Les données du problème (**P2**) sont les suivantes: ($n = 4$, $m = 10$, $Q = 40$)

arête $e \in E$	c_e
{0,1}	25
{0,2}	59
{0,3}	78
{0,4}	19
{1,2}	71
{1,3}	73
{1,4}	42
{2,3}	50
{2,4}	47
{3,4}	78

sommet $j \in V^*$	profit p_j	pénalité γ_j
1	16	8
2	19	9
3	23	11
4	21	10

Les données du problème (**P3**) sont les suivantes: ($n = 5$, $m = 15$, $Q = 45$)

arête $e \in E$	c_e
{0,1}	24
{0,2}	85
{0,3}	70
{0,4}	97
{0,5}	78
{1,2}	76
{1,3}	48
{1,4}	85
{1,5}	74
{2,3}	60
{2,4}	16
{2,5}	19
{3,4}	60
{3,5}	71
{4,5}	35

sommet $j \in V^*$	profit p_j	pénalité γ_j
1	25	12
2	24	12
3	15	7
4	15	7
5	11	5

Les données du problème (**P4**) sont les suivantes: ($n = 4$, $m = 10$, $Q = 46$)

arête $e \in E$	c_e
{0,1}	101
{0,2}	61
{0,3}	39
{0,4}	34
{1,2}	51
{1,3}	81
{1,4}	75
{2,3}	31
{2,4}	28
{3,4}	11

sommet $j \in V^*$	profit p_j	pénalité γ_j
1	24	12
2	21	10
3	23	11
4	23	11

Les données du problème (**P5**) sont les suivantes: ($n = 5$, $m = 15$, $Q = 44$)

arête $e \in E$	c_e
{0,1}	94
{0,2}	77
{0,3}	70
{0,4}	116
{0,5}	93
{1,2}	54
{1,3}	54
{1,4}	45
{1,5}	47
{2,3}	84
{2,4}	98
{2,5}	91
{3,4}	51
{3,5}	25
{4,5}	27

sommet $j \in V^*$	profit p_j	pénalité γ_j
1	22	11
2	13	6
3	25	12
4	10	5
5	18	9

Comparaison des relaxations linéaires des formulations EDCF et EUDCF

En considérant le problème (**P1**), on a trouvé que $Z_{LR-D} = Z_{LR-UD} = 67,6$.
En considérant le problème (**P2**), on a trouvé que $Z_{LR-D} = 69,5 > Z_{LR-UD} = 69,142$.
En considérant le problème (**P3**), on a trouvé que $Z_{LR-D} = 86,826 < Z_{LR-UD} = 89$.

On conclut que les relaxations linéaires des formulations basée sur les coupes EDCF et EUDCF sont incomparables.

Comparaison des relaxations des formulations EDCF et MSTF

En considérant le problème (**P4**), on a trouvé que $Z_{LR-D} = Z_{LGR} = 67$.
En considérant le problème (**P1**), on a trouvé que $Z_{LGR} = 94 > Z_{LR-D} = 67,6$.
En considérant le problème (**P5**), on a trouvé que $Z_{LGR} = 114,333 < Z_{LR-D} = 117,461$.

On conclut que les relaxations des formulations EDCF et MSTF sont incomparables.

Comparaison des relaxations des formulations EUDCF et MSTF

En considérant le problème (**P4**), on a trouvé que $Z_{LR-UD} = Z_{LGR} = 67$.
En considérant le problème (**P1**), on a trouvé que $Z_{LGR} = 94 > Z_{LR-UD} = 67,6$.
En considérant le problème (**P5**), on a trouvé que $Z_{LGR} = 114,333 < Z_{LR-D} = 117,461$.

On conclut que les relaxations des formulations EUDCF et MSTF sont incomparables.

5.6 Conclusion

Dans ce chapitre, nous avons développé une procédure exacte de type branch-and-cut pour résoudre le PCSTP qui est *NP*-difficile. Une formulation se basant sur les coupes (DCF) a été présentée. Cette dernière comporte

un nombre exponentiel de contraintes. Nous avons proposé d'ajouter des inégalités valides au modèle DCF pour améliorer son temps de convergence. Une procédure exacte de génération de coupes a été appliquée à ce modèle renforcé.

Avant de passer à la résolution exacte, nous avons développé un prétraitement efficace qui permet de réduire significativement la taille des instances. Ayant l'excellente borne supérieure obtenue grâce à l'heuristique HSPI, le preprocessing consiste à éliminer les arêtes et les sommets qui n'ont pas la possibilité d'appartenir à la solution optimale. Le preprocessing est constitué de deux étapes : la première étape se base sur la relaxation lagrangienne et la deuxième étape se base sur la relaxation linéaire et sur les coûts.

Nous présentons des résultats numériques qui montrent la pertinence de l'approche exacte mise en ouvre. Ces résultats montrent que des instances de grande taille sont résolues en un temps réduit. Cette partie a fait l'objet d'un article scientifique intitulé : "*An Exact Solution Approach for the Prize Collecting Steiner Tree Problem*" qui a été soumis en 2007 pour publication dans la revue internationale "*Discrete Optimization*" [103].

Chapitre 6

Développement de formulations compactes pour le PCSTP

6.1 Introduction

Depuis le développement de la programmation linéaire et des techniques d'optimisation combinatoire et le progrès parallèle des ordinateurs et des processeurs, les logiciels d'optimisation et les solveurs de programmes linéaires ont évolué avec une vitesse vertigineuse. En effet, des solveurs tel que CPLEX, LINDO etc... sont de plus en plus performants en terme de robustesse et aussi en temps de résolution. Ce qui justife notre objectif de développer des formulations mathématiques qu'on fournit directement à un solveur afin qu'il puisse la résoudre. Bien evidemment, il s'agit de formulations compactes avec un nombre polynômial de contraintes et de variables.

Comme pour la formulation DCF, à partir du graphe non orionté $G = (V, E)$, on construit un graphe orienté $B = (V, A)$ où l'ensemble des arcs A est obtenu en remplaçant toute arête $e = \{i, j\} \in E$ par deux arcs orientés (i, j) et (j, i) de même coût $c_{ij} = c_{ji} = c_e$. Notons que le noeud racine 0 n'admet pas d'arcs entrants. Ainsi, le PCSTP revient à chercher une arborescence optimale du graphe B dont la racine est le noeud 0.
Définissons les vecteurs des variables de décision $x \equiv (x_{ij})_{(i,j)\in A}$ et $y \equiv (y_j)_{j\in V}$. Avec, $x_{ij} = 1$ si l'arc $(i, j) \in A$ appartient à l'arborescence optimale, 0 sinon. Et, $y_j = 1$ si le noeud $j \in V$ appartient à l'arborescence optimale, 0 sinon.

En utilisant ces variables de décision, une formulation générique du PCSTP est la suivante:

$$\textbf{PCSTP}: \text{ Minimiser } \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{j\in V^*} \gamma_j(1-y_j) \tag{117}$$

sous contraintes:

$$y_1 = 1, \tag{118}$$

$$\sum_{j\in V^*} p_j y_j \geq Q, \tag{119}$$

$$\sum_{i\in\delta^-(j)} x_{ij} = y_j, \qquad \forall j \in V^*, \tag{120}$$

$$x \equiv (x_{ij})_{(i,j)\in A} \text{ définit une arborescence dont la racine est 0,} \tag{121}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A, \tag{122}$$

$$y_j \in \{0,1\}, \qquad \forall j \in V. \tag{123}$$

Il s'agit d'une nouvelle manière de modéliser le PCSTP. Notons que les contraintes (121) sont des contraintes d'élimination de sous-tours qu'on note généralement SEC (Subtour Elimination Constraints).

Dans ce chapitre, nous nous intéréssons au développement de contraintes SEC dont le nombre est polynomial. Nous commençons par présenter une première famille de formulations F1 inspirée des contraintes de type MTZ (Miller-Tucker-Zemlin [158]). Une deuxième famille de formulations F2 basée sur les chemins est aussi exposée. Nous développons aussi une troisième famille de formulations F3 basée sur la notion de flot. En plus, nous développons une comparaison théorique des relaxations linéaires de ces différentes formulations. Enfin, une étude expérimentale exhaustive comparant la performance de ces formulations en terme de relaxation linéaire, résolution exacte et approchée, et aussi de procédure de réduction de graphe, est illustrée à travers les tests numériques du paragraphe 7.6.

6.2 Formulations de type Miller-Tucker-Zemlin

Partant du modéle générique (117)-(123) du PCSTP, nous introduisons le vecteur de variables réelles $u \equiv (u_j)_{j\in V}$ qui est défini comme suit

$$\text{si } x_{ij} = 1, \text{ alors } u_i + 1 \leq u_j, \ \forall (i,j) \in A. \tag{124}$$

Les contraintes (124) ont été initialement introduites pour le problème de voyageur de commerce par Miller Tucker et Zemlin en 1960 [158]. Nous rappelons que le problème de voyageur de commerce (noté usuellement TSP) est un problème de référence en optimisation combinatoire. Etant données $n+1$ villes et les distances entre tout couple de villes, le problème du voyageur de commerce consiste a trouver un trajet de longueur minimale qui part nécessairement de la ville 0 et passe une et une seule fois par chaque ville i $(i = 1, ..., n)$ et revient à sa ville de départ. En d'autres termes, le TSP consiste a trouver un cycle hamiltonien de longueur minimum.

En 1960, Miller,Tucker et Zemlin étaient les premiers à écrire les contraintes (124) de manière linéaire. Il s'agit des contraintes, notées MTZ, qui suivent

$$u_i - u_j + 1 \leq M_{ij}(1 - x_{ij}),\ \forall (i,j) \in A, \tag{125}$$

où, pour tout arc $(i,j) \in A$, M_{ij} est un nombre assez grand. Généralement, on se contente de prendre $M_{ij} = n$, $\forall (i,j) \in A$. Reprenons l'exemple du problème de voyageur de commerce, si on fixe $u_0 = 0$ et on ajoute des contraintes sur les bornes telles que : $1 \leq u_j \leq n$, $\forall j \in V^*$, alors l'interprétation des variables u_i, $i = 0, ..., n$, est qu'ils représentent l'ordre/rang dans lequel les villes sont visitées. Les contraintes MTZ (125) peuvent être remplacées par les contraintes d'élimination sous-tours classiques et dont la formule est la suivante:

$$\sum_{(i,j) \in E(S)} x_{ij} \leq |S| - 1, \quad \forall S \subseteq V\ ,\ |S| \geq 2. \tag{126}$$

Ces contraintes (126) étaient introduites par Dantzig et al. en 1954 [43] pour le problème de voyageur de commerce avant d'être popularisées et largement appliquées pour les problèmes d'optimisation d'arbres. Ils sont d'un nombre exponentiel.

D'autre part, l'avantage majeur des contraintes MTZ est leur présentation polynomiale compacte puisque leur nombre est de l'ordre de $O(n^2)$. Cependant, il a été vérifié que ces contraintes produisent une assez faible relaxation linéaire (voir les travaux de Langevin et al. [135] et de Padberg et Sung [165]). Dans le paragraphe suivant, nous allons adapter et améliorer ces contraintes pour le problème PCSTP.

6.2.1 Formulation F1-MTZ

En 1995, Gouveia [87] a été le premier à adapter les contraintes MTZ classiques pour un problème d'optimisation arbre à savoir le problème d'arbre couvrant de poids minimal avec des contraintes supplémentaires. Il s'agit d'une restriction particulière limitant le nombre de connexions ou liens entre la racine et les feuilles de l'arborescence recherchée. Autre que cette application, à notre connaissance, ces contraintes MTZ ont été appliquées seulement pour deux versions du problème de steiner à travers les travaux de Voß [203] et Costa et.al [39].

Rappelons que pour chaque noeud $j \in V^*$, ϑ_j est le nombre minimal d'arcs dans le chemin du noeud racine 0 au noeud j. On fixe $\vartheta_0 = 0$. Nous commençons par définir les variables de decision $u_j \geq 0$, $j \in V$, qui représentent la position du noeud j dans l'arborescence. En d'autre termes, pour chaque noeud $j \in V^*$, u_j représente le nombre d'arcs du chemin reliant le noeud racine 0 au noeud j.

Pour le PCSTP, la contrainte (121) peut être remplacée par les contraintes de type MTZ qui suivent.

$$\begin{aligned}
(n+1-\vartheta_j)x_{ij} + u_i - u_j &\leq ny_i - \vartheta_j y_j, && \forall (i,j) \in A, && (127)\\
\vartheta_j y_j &\leq u_j \leq ny_j, && \forall j \in V^*, && (128)\\
u_0 &= 0. && && (129)
\end{aligned}$$

Ainsi, on btient la formulation F1-MTZ définie par:

$$\textbf{F1-MTZ:}\quad \text{Minimiser} \left\{ \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{j\in V^*} \gamma_j(1-y_j) : (118)\text{-}(120),\ (122)\text{-}(123) \text{ et } (127)\text{-}(129) \right\}.$$

Proposition 23 *Formulation F1-MTZ est une formulation valide pour le PCSTP.*

Preuve. Considérons une arborescence du graphe initial G dont la racine est 0, qui correspond à une solution réalisable du PCSTP selon le modèle (117)-(123). On définit la valeur de chaque variable u_j, $j = 1, ..., n$, par le nombre d'arcs dans le chemin du noeud racine 0 au noeud j. On fixe $u_0 = 0$. Ainsi, u_j ne peut pas dépasser le nombre total de noeud n et il est au moins

égale à ϑ_j, $j \in V^*$. Ce qui correspond à la contrainte (128).
Pour déduire la contrainte (127), il suffit de remarquer que pour tout arc $(i,j) \in A$, si $x_{ij} = 1$ alors $u_j = u_i + 1$. Ainsi, il s'agit bien d'une solution réalisable pour F1-MTZ.

Inversement, considérons une solution réalisable (**x**,**y**,**u**) de la formulation F1-MTZ. D'une part, la contrainte (120) guarantit que si un noeud $j \in V^*$ appartient à la solution alors, il existe un unique arc entrant à ce noeud. D'autre part, pour chaque arc $(i,j) \in A$, si $x_{ij} = 1$ alors on a $y_i = y_j = 1$ et par suite la contrainte (127) implique que $1 + u_i \leq u_j$. En sommant la contrainte (127) pour tous les arcs inclus dans un circuit donné, on obtient $u_i < u_i$ pour tout noeud i dans le circuit. Ce qui est absurde.
Ainsi, on conclut que toute solution réalisable du modèle F1-MTZ ne peut pas contenir de circuits. Ainsi, la solution (**x**,**y**) correspond à une arborescence réalisable du PCSTP selon le modèle (117)-(123). ■

Comme nous l'avons déjà présenté dans le paragraphe précédent, l'inconvenient majeur des contraintes MTZ est la faible performance de leur relaxation linéaire. Pour y remédier, Desrochers et Laporte [45] ont présenté une amélioration des contraintes (125) en appliquant le principe du "lifting". Dans le paragraphe suivant, nous présentons notre propore lifting de la formulation F1-MTZ qui a été inspiré des travaux de Desrochers et Laporte [45].

6.2.2 Formulation F1-DL

En 1991, trente ans après leur introduction par Miller, Tucker et Zemlin [158], les contraintes MTZ ont été reprises et revisitées par Desrochers et Laporte [45]. Ces derniers ont proposé d'ajouter un terme positif en fonction des variables à l'inégalité (125). Il s'agit de lifter les contraintes (125) par les contraintes suivantes:

$$M_{ij}x_{ij} + \alpha_{ji}x_{ji} + u_i - u_j \leq M_{ij},\ \forall (i,j),(j,i) \in A, \tag{130}$$

où, pour tout $(i,j),(j,i) \in A$, M_{ij} et α_{ji} sont deux réels positifs respectant les contraintes basiques (125).

En suivant l'approche de Desrochers et Laporte, nous liftons les contraintes (127) en proposant les contraintes (131) qui suivent

$$(n+1-\vartheta_j)x_{ij} + (n-1-\vartheta_j)x_{ji} + u_i - u_j \leq ny_i - \vartheta_j y_j,\ \forall (i,j),(j,i) \in A. \tag{131}$$

Proposition 24 *Les contraintes (131) sont valides pour le PCSTP.*

Preuve. Ce résultat se base sur le fait que pour une solution du PCSTP, il est impossible d'avoir à la fois $x_{ij} = 1$ et $x_{ji} = 1$, pour $(i, j), (j, i) \in A$. Considérons une solution réaslisable du PCSTP et deux arcs (i, j) et $(j, i) \in A$. Cherchons le plus grand réel positif α_{ji} tel que:
$(n + 1 - \vartheta_j)x_{ij} + \alpha_{ji}x_{ji} + u_i - u_j \leq ny_i - \vartheta_j y_j$
Deux cas peuvent se présenter:

- Si $x_{ji} = 0$, alors on retrouve la contrainte valide (127).

- Si $x_{ji} = 1$, alors on a nécessairement $x_{ij} = 0$, $y_i = y_j = 1$ et $u_i \geq u_j + 1$. Ainsi, $\alpha_{ji} \leq u_j - u_i + n - \vartheta_j \leq n - \vartheta_j - 1$.

Notons que $n - 1 - \vartheta_j$ est un terme toujours positif pour tout $j \in V$. Car, par définition, on a $1 \leq \vartheta_j \leq n$. Pour les noeuds $j \in V^*$ vérifiant $\vartheta_j = n$, on a nécéssairement $x_{ji} = 0, \forall i \in \delta^+(j)$. Donc, il suffit de prendre $\alpha_{ji} = n - 1 - \vartheta_j$. ■

Pour renforcer d'avantage la formulation F1-MTZ, on propose les contraintes (132) qui suivent

$$\begin{aligned} \vartheta_j y_j + \sum_{k \in \delta^-(j)} (\vartheta_k - \vartheta_j + 1)x_{kj} &\leq u_j \leq ny_j - (n-1)x_{0j}, \ \forall j \in V^*, \text{ si } 0 \in \delta^-(j) \\ \vartheta_j y_j + \sum_{k \in \delta^-(j)} (\vartheta_k - \vartheta_j + 1)x_{kj} &\leq u_j \leq ny_j, \ \forall j \in V^*, \text{ si } 0 \notin \delta^-(j) \end{aligned} \tag{132}$$

Proposition 25 *Les contraintes (132) sont valides pour le PCSTP.*

Preuve. Considérons une solution réaslisable du PCSTP et un noeud $j \in V^*$ vérifiant la contrainte (120). Deux cas peuvent se présenter:

- Si $y_j = 0$, alors $x_{kj} = 0$, $\forall\ k \in \delta^-(j)$ et par conséquent $u_j = 0$.

- Si $y_j = 1$ alors nécessairement il existe $k \in \delta^-(j)$ tel que $x_{kj} = 1$. Si $k = 0$, alors $\vartheta_j \leq u_j \leq 1$. Or, dans ce cas $\vartheta_j = 1$ puisque $0 \in \delta^-(j)$et par suite $u_j = 1$. Sinon, on a l'inégalité valide $1 + \vartheta_k \leq u_j \leq n$. Ainsi, les contraintes (132) sont vérifiées par toute solution du PCSTP. ■

Pour ce qui concerne le terme $\sum_{k \in \delta^-(j)} (\vartheta_k - \vartheta_j + 1)x_{kj}$ des contraintes (132), nous remarquons que pour tout noeud $j \in V^*$, on a toujours l'égalité:

$\sum_{k\in\delta^-(j)} (\vartheta_k - \vartheta_j + 1)x_{kj} = \sum_{k\in\delta^-(j),k\neq 0} (\vartheta_k - \vartheta_j + 1)x_{kj}$. En effet, il suffit de remarquer que $\vartheta_0 = 0$ et que si $0 \in \delta^-(j)$ alors $\vartheta_j = 1$.

Toujours à propos des contraintes (132), notons que pour tout noeud $j \in V^*$ et $k \in \delta^-(j)$, $\vartheta_k - \vartheta_j + 1$ est un terme positif. En effet, il est impossible d'avoir $\vartheta_j > \vartheta_k + 1$ par définition même de ϑ_j qui est le nombre minimal d'arcs reliant le noeud racine 0 au noeud j. Donc, les contraintes (132) sont bien un lifting des contraintes (128).

En conclusion, nous définissons la formulation F1-DL décrite comme suit

$$\textbf{F1-DL:}\quad \text{Minimiser} \left\{ \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{j\in V^*} \gamma_j(1-y_j) : (118)\text{-}(120),\ (122)\text{-}(123),\ (129) \text{ et } (131)\text{-}(132) \right\}.$$

6.2.3 Formulation F1-RLT

En 1990, la technique RLT (Reformulation Linearisation Technique) a été introduite par Sherali et Adams ([190], [191]). Il s'agit d'une technique d'optimisation nonconvexe globale dont l'idée de base est de reformuler un problème initial afin de générer des modèles avec d'excellentes relaxations linéaires. Le but est aussi de résoudre à l'optimalité des problèmes à variables entières.
En 2002, Sherali et Driscoll [196] ont présenté une application de cette technique au problème de voyageur de commerce. Ils ont démontré que la relaxation linéaire de leur formulation donne de meilleurs bornes inférieures que celle de la formulation avec les contraintes de Derocher et Laporte. Pour plus de détails sur le fondement théorique de cette technique ainsi que ses multiples applications, nous orientons le lecteur vers l'ouvrage de Sherali et Adams [189].

Comme le PCSTP est un problème combinatoire discret, nous avons adapté cette technique pour le modéliser. A notre connaissance, nous sommes les premiers à appliquer la RLT pour la modélisation des problèmes d'optimisation d'arbres.
Présentons une application de la technique RLT au problème PCSTP. En effet, on peut exprimer les contraintes d'élimination de sous-tours en utilisant

les contraintes non-linéaires suivantes:

$$\begin{aligned} u_j x_{ij} &= (u_i + 1)x_{ij}, && \forall (i,j) \in A, i \in V^*, j \in V^*, && (133)\\ u_j x_{0j} &= x_{0j}, && \forall j \in \delta^+(0), && (134)\\ \vartheta_j &\leq u_j \leq n, && \forall j \in V^*. && (135) \end{aligned}$$

Notons que si $y_j = 0$ alors u_j peut prendre toute valeur réelle dans l'intervalle $[\vartheta_j, n]$. Les contraintes non-linéaires (133)-(135) sont des contraintes d'élimination de sous-tours similaires aux contraintes (127). En effet, pour tout $(i,j) \in A$, si $x_{ij} = 1$ alors $u_j = u_i + 1$.

Comme son nom l'indique, la RLT se compose principalement de deux phases: la première phase consiste en la reformulation du problème suivi de la deuxième phase qui est la phase linéarisation. Lors de la phase de reformulation, des contraintes supplémentaires non-linéaires sont générées. L'application d'une linéarisation consiste à utilise une substitution des variables à la place de chacun des termes non-linéaires de la première phase. Des relations utiles entre les nouvelles variables et les variables originales sont aussi ajoutées afin de renforcer les liens de non-linéarité.

Phase de reformulation

(i) Partant de la contrainte (120), on construit la contrainte valide suivante:

$$\left[\sum_{i \in \delta^-(j)} x_{ij} = y_j \right] * u_j, \qquad \forall j \in V^*. \tag{136}$$

(ii) Considérons la contrainte (135), on obtient les deux contraintes qui suivent:

$$(\vartheta_j \leq u_j \leq n) * x_{ij}, \qquad \forall (i,j) \in A, i \in V^*, \tag{137}$$

$$(\vartheta_j \leq u_j \leq n) * y_j, \qquad \forall j \in V^*. \tag{138}$$

(iii) Une contrainte valide pour le PCSTP est la suivante : $x_{ij} + x_{ji} \leq y_j$, $(i,j),(j,i) \in A$ et $j \in V^*$. Il s'agit de la contrainte d'élimination de sous-tours basique. Il suffit de multiplier cette contrainte par les bornes supérieures et bornes inférieures de la contrainte (135), pour obtenir

$$(y_j - x_{ij} - x_{ji})\,(u_j - \vartheta_j) \geq 0, \qquad \forall (i,j),(j,i) \in A,\ j \in V^*, \tag{139}$$

$$(y_j - x_{ij} - x_{ji})\,(n - u_j) \geq 0, \qquad \forall (i,j),(j,i) \in A,\ j \in V^*. \tag{140}$$

$$(x_{ij} + x_{ji} \leq y_j) * u_j, \qquad \forall (i,j),(j,i) \in A,\ j \in V^*, \tag{141}$$

(vi) Tout noeud $j \in \delta^+(0)$ vérifie l'inégalité valide $x_{0j} \leq y_j$. Notons que cette inégalité n'est pas valide en égalité car on peut avoir $x_{0j} = 0$ et $y_j = 1$. Une inégalité non-linéaire est la suivante:

$$(y_j - x_{0j})\,(u_j - 2) \geq 0, \qquad \forall j \in \delta^+(0). \tag{142}$$

En effet, pour tout $j \in \delta^+(0)$, deux cas peuvent se présenter:

- Si $x_{0j} = 1$, alors $y_j = 1$, et par suite l'inégalité est vérifiée.
- Si $x_{0j} = 0$, alors si $y_j = 0$, et l'inégalité est vérifiée. Sinon ($y_j = 1$), alors nécessairement $u_j \geq 2$, d'où l'inégalité est vérifiée.

Comme pour tout noeud $j \in V^*$, on a $u_j \leq n$, en particulier pour tout noeud $j \in \delta^+(0)$, l'inégalité suivante est vérifiée:

$$(y_j - x_{0j})\,(n - u_j) \geq 0, \qquad \forall j \in \delta^+(0). \tag{143}$$

En plus, pour tout noeud $j \in \delta^+(0)$, l'inégalité non-linéaire suivante est vérifiée:

$$(x_{0j} \leq y_j) * u_j, \qquad \forall j \in \delta^+(0). \tag{144}$$

Notons que d'autres contraintes valides peuvent être utilisées lors de cette phase de reformulation. Cependant, il ne faut pas perdre de vue que chaque terme non-linéaire distinct produit lors de cette phase sera modélisé par une nouvelle variable lors de la phase de linéarisation, ce qui augmenterait la taille du modèle final. Les détails de cette linéarisation sont exposés dans le paragraphe qui suit.

Phase de linéarisation

Introduisons deux nouveaux vecteurs de variables $t \equiv (t_{ij})_{(i,j)\in A, i\in V^*}$ et $v \equiv (v_j)_{j\in V^*}$ afin de linéariser les nouvelles classes de contraintes (i)-(vi) en utilisant la substitution

$$t_{ij} = u_i x_{ij}, \quad \forall (i,j) \in A, i \in V^*, \ \text{ et } \ v_j = u_j y_j, \ \ \forall j \in V^*. \tag{145}$$

En plus, conséquence de la contrainte (134), on peut remplacer

$$u_i x_{0j} \quad \text{par} \quad x_{0j}, \quad \forall j \in V^*. \tag{146}$$

Reprenons la contrainte (133), la relation suivante est vérifiée:

$$u_j x_{ij} = t_{ij} + x_{ij}, (i,j) \in A, i \in V^*. \tag{147}$$

Par l'application de cette technique RLT, on obtient la formulation F1-RLT du PCSTP suivante:

$$\textbf{F1-RLT}: \quad \text{Minimiser} \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{j\in V^*} \gamma_j(1-y_j)$$

sous contraintes (118)-(120), (122)-(123),

$$\sum_{i\in\delta^-(j)} x_{ij} + \sum_{i\in\delta^-(j), i\neq 0} t_{ij} = v_j, \quad \forall j \in V^*, \tag{148}$$

$$\vartheta_i x_{ij} \leq t_{ij} \leq n x_{ij}, \ \forall (i,j) \in A, i \in V^*, \tag{149}$$

$$\vartheta_j y_j \leq v_j \leq n y_j, \quad \forall j \in V^*, \tag{150}$$

$$v_j - n y_j + (n-1)x_{ij} + n x_{ji} \leq t_{ij} + t_{ji} \leq$$

$$v_j - \vartheta_j y_j + (\vartheta_j - 1)x_{ij} + \vartheta_j x_{ji}, \ \forall j \in V^*, \ \forall (i,j),(j,i) \in A, j \in V^*, \tag{151}$$

$$x_{ij} + t_{ij} + t_{ji} \leq v_j, \ \forall (i,j),(j,i) \in A, j \in V^*, \tag{152}$$

$$2y_j - x_{0j} \leq v_j \leq n y_j - (n-1)x_{0j}, \quad \forall j \in \delta^+(0), \tag{153}$$

$$x_{0j} \leq v_j, \quad \forall j \in \delta^+(0), \tag{154}$$

$$t_{ij} \geq 0, \ \forall (i,j) \in A, i \in V^*, \ v_j \geq 0, \ \forall j \in V^*. \tag{155}$$

Grâce aux linéarisations (145) et (147) et la substitution (146), la contrainte (136) s'écrit sous la forme linéaire (148). Les contraintes (137) et (138) donnent lieu respectivement aux contraintes (149) et (150). Les bornes (inférieure et supérieure) de la contraintes (151) sont déduites respectivement des contraintes (139) et (140). De manière similaire, les contraintes (142) et (143) sont linéarisées en les bornes inférieure et supérieure de la contrainte (153). Enfin, les contraintes (152) et (154) sont obtenues respectivement à partir des contraintes (141) et (144).

Ainsi, on obtient une nouvelle formulation F1-RLT qui admet 4 vecteurs de variables ($\mathbf{x}$, $\mathbf{y}$, $\mathbf{t}$, $\mathbf{v}$) dont le nombre total est de l'ordre de $O(n^2)$.

Dans la section suivante, nous exposons une autre famille de formulations F2 basées sur la notion des chemins.

6.3 Formulations basées sur les chemins

Nous présentons une nouvelle manière de modéliser le PCSTP qui se base sur les chemins. Nous introduisons un nouveau vecteur de variables **z** défini par: $z_{ij} = 1$ s'il existe un chemin orienté dans l'arborescence optimale du noeud $i \in V$ vers le noeud $j \in V^* \setminus \{i\}$, et $z_{ij} = 0$ sinon. Ainsi, une contrainte valide d'élimination de sous-tours est la suivante:

$$z_{ij} + z_{ji} \leq y_j, \forall i, j \in V^* \times V^*, i \neq j. \tag{156}$$

En effet, si un noeud $j \in V^*$ est couvert par une solution du PCSTP ($y_j = 1$) alors, il n'existe pas à la fois un chemin de i vers j et aussi un chemin de j vers i, $i \in V^*$, i.e, il n'existe pas de circuit contenant ce noeud j. Dans les paragraphes qui suivent, nous proposons une famille de formulations se basant sur ces contraintes (156).

6.3.1 Formulation F2-P

Afin de renforcer la formulation, nous ajoutons les contraintes valides qui suivent

$$\begin{aligned}
x_{ij} &\leq z_{ij}, && \forall (i,j) \in A, && (157)\\
z_{ik} + 1 &\geq z_{ij} + z_{jk}, && \forall i \in V, j, k \in V^*, i \neq j \neq k. && (158)\\
z_{ij} &\in \{0,1\} && \forall\ i \in V, j \in V^* \setminus \{i\}. && (159)
\end{aligned}$$

Il est clair que si un arc $(i,j) \in A$ est couvert par la solution ($x_{ij} = 1$), alors il existe naturellement un chemin de i vers j ($z_{ij} = 1$), d'où la contrainte (157). La contrainte (158) exprime le fait que pour un triplet $(i,j,k) \in V \times V^* \times V^*$ $i \neq j \neq k$, s'il existe un chemin de i vers j ($z_{ij} = 1$) et s'il existe un chemin de j vers k ($z_{jk} = 1$) alors nécessairement il existe un chemin de i vers k ($z_{ik} = 1$). La contrainte (159) traduit le fait que les variables du vecteur **z** sont de type binaire même si avec la contrainte (156), on a toujours z_{ij} dans $[0,1]$.

Remarque 2 *Un type similaire de variables a été utilisé par Gouveia et Pires [90] pour la formulation du problème du voyageur de commerce. En effet, ils ont proposé des formes différentes des contraintes respectives (158)*

et (156)

$$z_{ik} + 1 \geq z_{ij} + x_{jk},\ \forall (j,k) \in A, j \in V^*,\ \forall i \in V, i \neq j \neq k, \quad (160)$$
$$z_{jk} + x_{kj} \leq 1,\ \ \forall (k,j) \in A,\ k \in V^*. \quad (161)$$

En appliquant la contrainte (157), on trouve que les contraintes (158) et (156) que nous proposons sont potentiellement plus fortes et dominent les contraintes (160) et (161).

Remarque 3 *Toujours pour le problème de voyageur du commerce, Sherali et al. [197] ont proposé les contraintes qui suivent*

$$z_{ij} - z_{kj} \geq -(1 - x_{ik}),\ \forall (i,k) \in A,\ \forall j \in V^*, i \neq j \neq k, \quad (162)$$
$$z_{ij} - z_{kj} \leq 1 - x_{ik},\ \forall (i,k) \in A,\ \forall j \in V^*, i \neq j \neq k. \quad (163)$$

Les contraintes (162)-(163) supposent que pour un arc $(i,k) \in A$, si $x_{ik} = 1$, alors nécessairement, on a l'égalité $z_{ij} = z_{kj}$, $j \in V^*$. Or, cette égalité n'est pas valide pour l'arborescence optimale du PCSTP. En effet, on peut avoir le cas $x_{ik} = 1$ et $x_{ij} = 1$, alors $z_{ij} = 1$, mais il n'est pas nécessaire que $z_{kj} = 1$ (voir exemple de la figure 7.1 ci-dessous). Cependant, la contrainte (163)

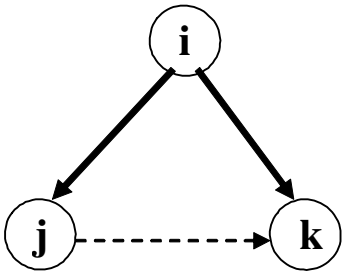

Figure 6.1: Contre-exemple des contraintes (162)-(163) de Sherali et al. [197]

est équivalente à la contrainte (160) de Gouveia et Pires [90] dominée par la contrainte (158) que nous proposons.

Les contraintes valides suivantes peuvent être ajoutées pour renforcer la formulation proposée:

$$z_{0j} = y_j, \qquad \forall j \in V^*, \quad (164)$$

$$\sum_{k\in V^*,k\neq j} z_{jk} \leq (n-\vartheta_j-1) \sum_{k\in\delta^+(j)} x_{jk}, \qquad \forall j \in V^*. \tag{165}$$

La contrainte (164) exprime le fait que si un noeud j, $j \in V^*$, appartient à la solution alors il existe évidemment un chemin du noeud racine 0 vers le noeud j. La contrainte (165) indique que si un noeud j, $j \in V^*$, est une feuille de l'arborescence solution ($\sum_{k\in\delta^+(j)} x_{jk} = 0$), alors nécessairement j n'admet pas de successeurs ($\sum_{k\in V^*,k\neq j} z_{jk} = 0$).

Pour développer des nouvelles contraintes valides pour le modèle, nous nous sommes inspirés des variables **u** de la famille de formulations F1. En effet, comme la valeur u_j, $j \in V^*$, peut être interprété comme étant le nombre des prédecesseurs du noeud j, on peut déduire la relation suivante:

$$u_j = \sum_{i\in V,i\neq j} z_{ij}, \qquad \forall j \in V^*. \tag{166}$$

Ainsi, similaires aux contraintes (132) et (131) que nous avons démontré valides pour le PCSTP, on introduit les contraintes qui suivent

$$\begin{aligned} \vartheta_j y_j + \sum_{k\in\delta^-(j)} (\vartheta_k - \vartheta_j + 1)x_{kj} &\leq \sum_{i\in V,i\neq j} z_{ij} \leq n y_j - (n-1)x_{0j}, \\ \forall j &\in V^*, 0 \in \delta^-(j) \\ \vartheta_j y_j + \sum_{k\in\delta^-(j)} (\vartheta_k - \vartheta_j + 1)x_{kj} &\leq \sum_{i\in V,i\neq j} z_{ij} \leq n y_j, \ \forall j \in V^*, \\ 0 &\notin \delta^-(j) \end{aligned} \tag{167}$$

$$\begin{aligned} & (n+1-\vartheta_j)x_{ij} + (n-\vartheta_j-1)x_{ji} + \sum_{k\in V,k\neq i} z_{ki} - \sum_{k\in V,k\neq j} z_{kj} \\ \leq\ & n y_i - \vartheta_j y_j, \ \forall (i,j),(j,i) \in A, i \neq 0 \end{aligned} \tag{168}$$

Il est clair que pour un arc (i,j), $(i,j) \in A$, si l'arc (j,i) n'existe pas alors la contrainte (168) s'écrit de la forme suivante:

$$(n+1-\vartheta_j)x_{ij} + \sum_{k\in V,k\neq i} z_{ki} - \sum_{k\in V,k\neq j} z_{kj} \leq n y_i - \vartheta_j y_j, \qquad \forall (i,j) \in A, i \neq 0.$$

Ainsi, nous définissons la formulation F2-P modélisée comme suit

$$\textbf{F2-P:}\quad \text{Minimiser} \left\{ \begin{array}{c} \sum\limits_{(i,j)\in A} c_{ij}x_{ij} + \sum\limits_{j\in V^*} \gamma_j(1-y_j) : \text{(118)-(120), (122)-(123),} \\ \text{(156)-(159), (164)-(165) et (167)-(168)} \end{array} \right\}.$$

Il s'agit de la formulation de base pour la famille de formulations F2. Dans le paragraphe suivant, nous présentons trois nouvelles formulations en appliquant le principe du lifting.

6.3.2 Formulations F2-PL

Pour le problème du voyageur de commerce, Gouveia et Pires [90] ont proposé le lifting suivant de la contrainte (160):

$$z_{ik} + 1 \geq z_{ij} + x_{jk} + x_{ik} + x_{ji}, \qquad \forall (j,k),(i,k),(j,i) \in A, j \in V^*. \quad (169)$$

Cette contrainte (169) n'est pas valide pour l'arborescence recherchée du PCSTP. En effet, considérons les arcs $(j,k),(i,k),(j,i) \in A$, si $x_{ji} = 1$ et $x_{jk} = 1$, alors pour ne pas avoir de circuit, on a $z_{ij} = 0$ et $x_{ik} = 0$. La contrainte (169) induit que $z_{ik} = 1$. Ce qui n'est pas valide pour une solution du PCSTP, car sous ces hypothèse il ne faut pas avoir de chemin de i vers k ($z_{ik} = 0$). Dans le même papier [90], Gouveia et Pires ont proposé un autre lifting de la contrainte (160) comme suit

$$z_{ik} + 1 \geq z_{ij} + x_{jk} + x_{kj}, \qquad \forall (j,k),(k,j) \in A,\ \forall i \in V. \quad (170)$$

Nous remarquons que comme tout arc (j,k), $(j,k) \in A$, vérifie l'inégalité $z_{jk} \geq x_{jk}$, alors la contrainte (169) domine potentiellement la contrainte (170).

Notre objectif est de lifter les contraintes (158) afin de dériver de nouvelles formulations plus performante que les formulations F2-P. Une première possibilité de lifting des contraintes (158) se présente à travers la contrainte qui suit

$$z_{ik} + 1 \geq z_{ij} + z_{jk} + x_{ik}, \qquad \forall (i,k) \in A,\ \forall j \in V^*, j \neq i \neq k. \quad (171)$$

Proposition 26 *La contrainte (171) est valide pour le problème PCSTP.*

Preuve. Partant d'une solution (**x,y,z**) du PCSTP. Considérons un arc (i,k), $(i,k) \in A$, si $x_{ik} = 0$, alors on retrouve la contrainte (158).

Supposons que $x_{ik} = 1$, alors on a évidemment $z_{ik} = 1$. Soit un noeud j, $j \in V^*$ tel que $j \neq i \neq k$. S'il existe un chemin de j vers k ($z_{jk} = 1$) alors il existe un chemin de j vers i ($z_{ji} = 1$) car le noeud k vérifie la contrainte (120). Et par suite, il n'existe pas de chemin de i vers j ($z_{ij} = 0$). Ainsi, l'inégallité (171) est vérifiée.

En plus, s'il existe un chemin de i vers j ($z_{ij} = 1$), alors il n'existe pas de chemin de j vers k ($z_{jk} = 0$) car il n'y a pas de circuit dans toute arborescence du PCSTP. On conclut que la contrainte (171) est valide pour le PCSTP. ■

On obtient ainsi la formulation F2-PL0 définie par

$$\textbf{F2-PL0:}\quad \text{Minimiser} \left\{ \begin{array}{l} \sum\limits_{(i,j)\in A} c_{ij}x_{ij} + \sum\limits_{j\in V^*} \gamma_j(1-y_j) : \text{(118)-(120), (122)-(123),} \\ \text{(156)-(157), (159), (164)-(165), (167)-(168) et (171)} \end{array} \right\}.$$

Nous proposons une deuxième stratégie de lifting des contraintes (158) à travers la contrainte suivante:

$$z_{ik} + 1 \geq z_{ij} + z_{jk} + x_{kj}, \qquad \forall (k,j) \in A, k \in V^*, \forall i \in V, i \neq j \neq k. \quad (172)$$

Proposition 27 *La contrainte (172) est valide pour le problème PCSTP.*

Preuve. Partant d'une solution (**x,y,z**) du PCSTP. Considérons un arc (k,j), $(k,j) \in A$, si $x_{kj} = 0$, alors on retrouve la contrainte (158).

Supposons que $x_{kj} = 1$, alors il n'existe pas de chemin de j vers k ($z_{jk} = 0$). Pour tout noeud i, $i \in V$ avec $i \neq j \neq k$, supposons que $z_{ij} = 1$ i.e il existe un chemin de i vers j. Or, k est un prédcesseur de j, alors on a nécessairement $z_{ik} = 1$. Ainsi, la contrainte (172) est valide pour le PCSTP. ■

On obtient ainsi la formulation F2_PL1 définie comme suit

$$\textbf{F2-PL1:}\quad \text{Minimiser} \left\{ \begin{array}{l} \sum\limits_{(i,j)\in A} c_{ij}x_{ij} + \sum\limits_{j\in V^*} \gamma_j(1-y_j) : \text{(118)-(120), (122)-(123),} \\ \text{(156)-(157), (159), (164)-(165), (167)-(168) et (172)} \end{array} \right\}.$$

Nous proposons de considérer une troisième formulation regroupant les deux contraintes liftées (169) et (170). Il s'agit de la formulation F2_PL2

suivante:

$$\textbf{F2-PL2:}\quad \text{Minimiser} \left\{ \begin{array}{c} \sum\limits_{(i,j)\in A} c_{ij}x_{ij} + \sum\limits_{j\in V^*} \gamma_j(1-y_j) : \text{(118)-(120), (122)-(123),} \\ \text{(156)-(157), (159), (164)-(165), (167)-(168) et (171)-(172)} \end{array} \right\}.$$

Comme pour la famille de formulations F1, nous appliquons la technique RLT à cette formulations basée sur les chemins dans le paragraphe qui suit.

6.3.3 Formulation F2-RLT

A notre connaissance, nous sommes les premiers à appliquer la technique RLT sur ce genre de formulation basées sur les chemins.
Nous commençons par générer des contraintes supplémentaires non-linéaires dans la première phase de reformulation. Par l'application de la deuxième phase de linéarisation, nous substituons chacun des termes non-linéaires par de nouvelles variables. Bien évidemment, nous tenons compte des relations utiles entre les nouvelles variables et les variables initiales afin de renforcer le modèle recherché.

Phase de reformulation

Commençons par les contraintes valides suivantes:

$$(0 \leq x_{ik} \leq 1) * z_{kj}, \qquad \forall i \in V, \forall j,k \in V^*, k \in \delta^+(i), k \neq j, \tag{173}$$

$$(0 \leq z_{kj} \leq 1) * x_{ik}, \qquad \forall i \in V, \forall j,k \in V^*, k \in \delta^+(i), k \neq j, \tag{174}$$

$$(0 \leq z_{ik} \leq 1) * z_{kj}, \qquad \forall i,j,k \in V \times V^* \times V^*, i \neq j \neq k, \tag{175}$$

Partant de la contrainte (157), on construit la contrainte valide suivant:

$$(x_{ij} \leq z_{ij}) * z_{jk}, \qquad \forall (i,j) \in A, \forall k \in V^*, k \neq i \neq j. \tag{176}$$

Considérons la contrainte (120), on obtient la contrainte non-linéaire qui suit

$$\left[\sum_{i\in\delta^-(j)} x_{ij} = y_j \right] * z_{jk}, \qquad \forall j,k \in V^*,\ j \neq k. \tag{177}$$

A partir de la contrainte (172) que nous avons montré valide pour le PCSTP, on écrit la contrainte suivante:

$$(z_{ik} + 1 \geq z_{ij} + z_{jk} + x_{kj}) * z_{jk}, \quad \forall (k,j) \in A, k \in V^*, \forall i \in V, i \neq j \neq k. \tag{178}$$

Toutes ces contraintes sont non-linéaires, il faut passer à la phase de linéarisation pour obtenir un modèle linéaire du PCSTP.

Phase de linéarisation

Nous commençons par introduire les nouveaux vecteurs de variables **h** et **l** définis par:

$$h_{ij}^k = x_{ik}z_{kj},\ \forall i \in V, \forall j,k \in V^*, k \in \delta^+(i), k \neq j, \tag{179}$$

$$l_{ij}^k = z_{ik}z_{kj},\ \forall i \in V, \forall j,k \in V^*, k \neq j \neq i. \tag{180}$$

Il est évident que si un noeud j, $j \in V^*$, n'est pas couvert par la solution ($y_j = 0$) alors il n'existe aucun chemin de j vers k, i.e $z_{jk} = 0$, $k \in V^*, k \neq j$. Cette relation valide s'écrit comme suit

$$z_{jk}y_j = z_{jk},\ \forall j,k \in V^*, k \neq j, \tag{181}$$

Pour tout arc (k,j), $(k,j) \in A$, il est impossible d'avoir à la fois $x_{kj} = 1$ et $z_{jk} = 1$. Sinon, il existe un circuit contenant l'arc (k,j) dans la solution du PCSTP. Ainsi, on obtient l'égalité valide suivante:

$$x_{kj}z_{jk} = 0,\ \forall (k,j) \in A, k \in V^*, \tag{182}$$

Suite à la contrainte (159) qui exprime le fait que les variables $(z_{jk})_{j,k \in V^*, j \neq k}$ sont binaires, l'égalité suivante est valide:

$$z_{jk}^2 = z_{jk},\ \forall j,k \in V^*, j \neq k. \tag{183}$$

Ainsi, grâce à la technique RLT, on obtient la formulation F2-RLT du PCSTP suivante:

$$\textbf{F2-RLT}: \quad \text{Minimiser} \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{j \in V^*} \gamma_j(1 - y_j)$$

sous contraintes (118)-(120), (122)-(123), (156)-(159), (164)-(165), (167)-(168),

$$h_{ik}^j \leq l_{ik}^j,\ \forall (i,j) \in A, \forall k \in V^*, i \neq j \neq k, \tag{184}$$

$$\sum_{i \in \delta^-(j)} h_{ik}^j = z_{jk},\quad \forall j,k \in V^*, j \neq k, \tag{185}$$

$$l_{ik}^j + l_{ij}^k \leq z_{ik},\ \forall i \in V, \forall j,k \in V^*, i \neq j \neq k, \tag{186}$$

$$x_{ij} + \sum_{k \in \delta^+(i), k \neq j (k \neq 1)} h_{ij}^k \geq z_{ij},\quad \forall (i,j) \in A, \tag{187}$$

$$0 \leq h_{ij}^k \leq z_{kj}, \ \forall i \in V, \forall j, k \in V^*, k \in \delta^+(i), k \neq j, \tag{188}$$

$$0 \leq h_{ij}^k \leq x_{ik}, \ \forall i \in V, \forall j, k \in V^*, k \in \delta^+(i), k \neq j, \tag{189}$$

$$0 \leq l_{ij}^k \leq z_{kj}, \ \forall i \in V, \forall j, k \in V^*, i \neq j \neq k. \tag{190}$$

Grâce à une simple substitution par les relations (179)-(180), les contraintes non-linéaires (173)-(175) et (176) donnent respectivement les contraintes (188)-(190) et (184). Nous linéarisons la contrainte (177) en la contrainte (185) par la substitution des termes non-linéaires en appliquant les relations (179) et (181). En plus, la contrainte (186) représente le résultat de la linéarisation de la contrainte (178) en utilisant successivement les relations (180), (182), (183) et aussi l'inégalité valide $z_{jk} \leq 1 - z_{kj}$, $\forall j, k \in V^* j \neq k$. Pour ce qui concerne la contrainte (187), il s'agit de la condition valide suivante: Pour tout arc (i, j), $(i, j) \in A$, si $z_{ij} = 1$ alors, on a nécéssairement soit $x_{ij} = 1$, soit $\sum_{k \in \delta^+(i)} h_{ij}^k = 1$.

Notons que pour tout triplet $(i, j, k) \in V \times V^* \times V^*$ $i \neq j \neq k$, s'il n'existe pas de chemin de i vers j ($z_{ij} = 0$), alors il est impossible d'avoir à la fois un chemin de i vers k et aussi un chemin de k vers j ($z_{ik} = z_{kj} = 1$). En d'autres termes, si $z_{ij} = 0$ alors $z_{ik} z_{kj} = 0$. Il s'agit de la contrainte valide qui suit

$$l_{ij}^k \leq z_{ik}, \qquad \forall i, j, k \in V, i \neq k \neq j \neq 1. \tag{191}$$

Nous n'avons pas considéré cette contrainte (191) dans le modèle F2-RLT puisqu'elle est dominée par la contrainte (186).

Ainsi, nous avons développé plusieurs variantes de la famille de formulations F2 qui se base sur les chemins. Dans la section suivante, nous présentons une troisième famille de formlations F3 du PCSTP qui se base sur la notion du flot.

6.4 Formulations basées sur les flots

Une manière classique d'aborder les problèmes d'optimisation d'arbres est de les formuler comme des problèmes particuliers de flot. Dans cette section, nous allons explorer cette approche pour le PCSTP.

6.4.1 Formulation multiflot F3-MC

Introduisons d'abord une illustration du problème de multiflot, où plusieurs types de produits doivent être transportés des sources de ces produits à des destinations précises, à travers un réseau. Prenons l'exemple de réseau routier. Chaque route a une capacité de transport, et il y a des demandes associées à chaque paire de source-destination. Le réseau qui connecte les sources aux destinations peut être modélisé comme un graphe. Le probléme consiste donc à trouver un ensemble de routes entre les sources et les destinations satisfaisant leurs demandes respectives. Quand il s'agit d'un seule type de produit et d'une seule source, le problème est facile à résoudre [70]. Sauf que, déjà avec deux types de produits, ce problème est NP-complet [60].

Le problème de l'arbre de Steiner (STP) a déjà été considéré comme un problème de multiflot par Wong [209]. Dans ce même papier consacré au STP, Wong a aussi démontré que les relaxations linéaires de la formulation multiflot et de celle basée sur les coupes directes sont équivalentes.

Pour le PCSTP, nous proposons la formulation F3-MC suivante:

$$\textbf{F3-MC}: \quad \text{Minimiser} \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{j\in V^*} \gamma_j(1-y_j)$$

sous contraintes (118)-(120), (122)-(123),

$$\sum_{j\in\delta^+(0)} f^k_{0j} = y_k, \qquad \forall k \in V^*, \tag{192}$$

$$\sum_{i\in\delta^-(k)} f^k_{ik} = y_k, \qquad \forall k \in V^*, \tag{193}$$

$$\sum_{i\in\delta^-(j)} f^k_{ij} - \sum_{i\in\delta^+(j)} f^k_{ji} = 0, \qquad \forall k \in V^*, \forall j \in V^*, j \neq k, \tag{194}$$

$$0 \leq f^k_{ij} \leq x_{ij}, \qquad \forall (i,j) \in A, \forall k \in V^*. \tag{195}$$

En effet, toute variable f^k_{ij}, $(i,j) \in A, k \in V^*$, correspond à la quantité de flot de type/commodité k circulant sur l'arc (i,j). A chaque arc $(i,j) \in A$ est associé une capacité x_{ij}, d'où la contrainte (195). Les contraintes (192) et (193) garantissent que pour chaque noeud k, $k \in V^*$, il existe un flot de type k et de quantité y_k, circulant de la source 0 vers la destination k. En

d'autres termes, la demande associée à chaque noeud k, $k \in V^*$, est égale à y_k. La contrainte (194) traduit le principe classique de conservation de flot.

En conclusion, ces contraintes (192)-(195) garantissent que dans tout sous-ensemble d'arcs correspondant à une solution réalisable, il existe un chemin du noeud racine 0 vers tout noeud k, $k \in V^*$, couvert par la solution du PCSTP. La formulation multiflot admet un nombre de variables de l'ordre de $o(n^3)$.

6.4.2 Formulation F3-SC

Partant de la formulation F3-MC, nous formulons le PCSTP comme un problème particulier de flot (*Single-Commodity Flow Problem SCF*) . En effet, on peut dériver un modèle plus compact que celui du F3-MC en introduisant les variables $f_{ij} = \sum_{k \in V^*} f_{ij}^k = \sum_{k \in V^*, k \neq i} f_{ij}^k \ \ \forall (i,j) \in A$, car $f_{ij}^i = 0$, $\forall (i,j) \in A$, $i \neq 0$.

Partant des contraintes (192)-(195), nous déduisons les contraintes qui suivent

$$\sum_{j \in \delta^+(0)} f_{0j} = \sum_{k \in V^*} y_k, \tag{196}$$

$$\sum_{i \in \delta^-(k)} f_{ik} - \sum_{i \in \delta^+(k)} f_{ki} = y_k, \qquad \forall k \in V^*, \tag{197}$$

$$x_{ij} \leq f_{ij} \leq (n - \vartheta_i) x_{ij}, \qquad \forall (i,j) \in A. \tag{198}$$

La contrainte (197) est obtenu en sommant sur k, $k = 1, ..., n$, les contraintes (193) avec la contrainte (194). La somme des contraintes (192) pour k, $k = 1, ..., n$, donne la contrainte (196). En appliquant le même principe sur la contrainte (195), on obtient:

$$0 \leq f_{ij} \leq n x_{ij}, \ \forall (i,j) \in A. \tag{199}$$

Or, nous proposons la contrainte (198) qui domine cette contrainte (199). En effet, si un arc $(i,j) \in A$ n'est pas couvert par la solution ($x_{ij} = 0$), alors aucune unité de flot ne peut ciculer sur cet arc, i.e $f_{ij} = 0$. D'autre part, si $x_{ij} = 1$, alors la quantité de flot circulant sur l'arc (i,j) ne peut pas dépasser la quantité $n - \vartheta_i$ suite aux contraintes (196)-(197).

Introduisons pour chaque noeud j, $j \in V$, la variable $s_j = \sum_{k \in V^*, k \neq j} z_{jk} = \sum_{k \in \delta^+(j)} f_{jk}$. Une étroite relation existe entre les valeurs des variables $(u_j)_{j \in V^*}$, $(z_{ij})_{i,j \in V^*, i \neq j}$ et $(s_j)_{j \in V}$ des formulations respectives F1, F2 et F3. Similaire à la valeur $u_j = \sum_{i \in V} z_{ij}$ de la formulation F1-MTZ qui représente le nombre de prédecesseurs du noeud $j \in V^*$, nous considérons la variable s_j qui représente le nombre de successeurs du noeud $j \in V$ dont la formule est la suivante:

$$s_j = \sum_{k \in \delta^+(j)} f_{jk}, \ \forall j \in V. \tag{200}$$

Par conséquent, on déduit l'inégalité valide qui suit

$$0 \leq s_j \leq (n - \vartheta_j) y_j \ \forall j \in V. \tag{201}$$

En effet, pour la racine 0, le nombre de successeur s_0 ne peut pas dépasser n. Pour un noeud $j \in V^*$, si $y_j = 0$, alors on a $s_j = 0$. Sinon, on trouve que $s_j \leq n - \vartheta_j$. Par suite, on peut substituer la valeur s_j, $j \in V$, par sa formule (200) en fonction des variables **f,** dans la contrainte (201). C'est ainsi qu'on obtient la contrainte qui suit

$$\sum_{k \in \delta^+(j)} f_{jk} \leq (n - \vartheta_j) y_j, \qquad \forall j \in V. \tag{202}$$

En conclusion, nous présentons la formulation F3-SC définie par

$$\textbf{F3-SC:} \ \text{Minimiser} \left\{ \begin{array}{c} \sum_{(i,j) \in A} c_{ij} x_{ij} + \sum_{j \in V^*} \gamma_j (1 - y_j) : (118)\text{-}(120), \ (122)\text{-}(123), \\ (196)\text{-}(198) \text{ et } (202) \end{array} \right\}.$$

6.4.3 Formulation F3-SC-MTZ

A partir de l'interprétation des variables $(s_j)_{j \in V^*}$, nous avons remarqué la relation suivante:

$$\text{Si } x_{ij} = 1, \text{ alors } s_j \leq s_i - 1, \ \forall (i,j) \in A. \tag{203}$$

Similaires aux contraintes (124) de Miller Tucker et Zemlin [158], il s'agit de contraintes d'élimination de sous-tours non-linéaires. Ainsi, en appliquant l'égalité (200), nous déduisons une nouvelle contrainte similaire aux contraintes (127) dont la formule est la suivante:

$$(n + 1 - \vartheta_j) x_{ij} + \sum_{k \in \delta^+(j)} f_{jk} - \sum_{k \in \delta^+(i)} f_{ik} \leq (n - \vartheta_j) y_j, \qquad \forall (i,j) \in A. \tag{204}$$

Proposition 28 *La contrainte (204) est valide pour le PCSTP.*

Preuve. Soit un arc $(i,j) \in A$, cherchons un réél positif M_{ij} vérifiant l'inégalité $M_{ij}x_{ij} + s_j - s_i \leq (n - \vartheta_j)y_j$. Deux cas peuvent se présenter:

- Si $x_{ij} = 0$, alors d'après la contrainte (201), $s_j \leq (n - \vartheta_j)y_j$ et $-s_i \leq 0$. Par conséquent, la contrainte (204) est valide.

- Si $x_{ij} = 1$, alors on a nécessairement $y_j = y_i = 1$ et $s_j - s_i \leq -1$, suite à la relation (203). Ainsi, on a $M_{ij} + s_j - s_i \leq -1 + M_{ij}$. En plus, il est impératif d'avoir $M_{ij} + s_j - s_i \leq n - \vartheta_j$. Alors, il faut choisir un réel M_{ij} tel que $M_{ij} \leq n + 1 - \vartheta_j$. Donc, il suffit de fixer $M_{ij} = n + 1 - \vartheta_j$, $\forall (i,j) \in A$. ■

Nous considérons une nouvelle formulation du PCSTP qu'on note F3-SC-MTZ qui est définie par

$$\textbf{F3-SC-MTZ:}\quad \text{Minimiser} \left\{ \begin{array}{l} \sum\limits_{(i,j)\in A} c_{ij}x_{ij} + \sum\limits_{j\in V^*} \gamma_j(1-y_j) : (118)\text{-}(120), \\ (122)\text{-}(123),\ (196)\text{-}(198),\ (202) \text{ et } (204) \end{array} \right\}.$$

Nous attirons l'attention du lecteur que cette dernière contrainte (204) ne peut pas être liftée par la contrainte qui suit

$$(n+2-\vartheta_j)x_{ij} + \sum_{k\in\delta^-(j),k\neq i} x_{kj} + \sum_{k\in\delta^+(j)} f_{jk} - \sum_{k\in\delta^+(i)} f_{ik} \leq (n+1-\vartheta_j)y_j, \forall (i,j) \in A. \tag{205}$$

En effet, la contrainte (205) n'est pas un lifting de la contrainte (204) car elle peut s'écrire de la manière suivante:

$(n+1-\vartheta_j)x_{ij} + x_{ij} + \sum\limits_{k\in\delta^-(j),k\neq i} x_{kj} + \sum\limits_{k\in\delta^+(j)} f_{jk} - \sum\limits_{k\in\delta^+(i)} f_{ik}$

$\leq (n+1-\vartheta_j)y_j$, $\forall (i,j) \in A$

$\Rightarrow (n+1-\vartheta_j)x_{ij} + \sum\limits_{k\in\delta^-(j)} x_{kj} + \sum\limits_{k\in\delta^+(j)} f_{jk} - \sum\limits_{k\in\delta^+(i)} f_{ik} \leq (n+1-\vartheta_j)y_j$,

$\forall (i,j) \in A$. Or, d'après la contrainte (120), on a l'agalité $\sum\limits_{k\in\delta^-(j)} x_{kj} = y_j$, $j \in V^*$. Et par suite, on obtient la contrainte (204).

Dans le paragraphe qui suit, nous présentons une nouvelle formulation du PCSTP basée sur la notion du flot et sur la technique RLT.

6.4.4 Formulation F3-SC-RLT

Comme pour les familles de formulations F1 et F2, nous appliquons la technique RLT sur la famille de formulation F3. En effet, les contraintes

d'élimination de sous-tour (203) peuvent être exprimées en utilisant les contraintes non-linéaires qui suivent

$$s_j x_{ij} \leq (s_i - 1)x_{ij}, \ \forall (i,j) \in A, \tag{206}$$

Phase de reformulation Un ensemble de contraintes supplémentaire est présenté lors de cette première phase de la technique RLT. Commençons par la contrainte non-linéaire valide suivante:

$$s_i = \sum_{j \in \delta^+(i)} s_j x_{ij} + \sum_{j \in \delta^+(i)} x_{ij}, \ \forall i \in V. \tag{207}$$

En plus, d'après la contrainte (196), on a $s_0 = \sum_{j \in V^*} y_j$. Dans ce cas, la contrainte (207) s'écrit comme suit

$$\sum_{j \in V^*} y_j = \sum_{j \in \delta^+(0)} s_j x_{0j} + \sum_{j \in \delta^+(0)} x_{0j}. \tag{208}$$

Passons à la contrainte valide classique: $x_{ij} + x_{ji} \leq y_j, j \in V^* \, (i,j), (j,i) \in A$, qu'on multiplie par s_j afin d'obtenir la contrainte suivante:

$$(x_{ij} + x_{ji} \leq y_j) * s_j, \forall j \in V^*. \tag{209}$$

En faisant appel à la contrainte (120), on a

$$\left[\sum_{i \in \delta^-(j)} x_{ij} = y_j \right] * s_j, \quad \forall j \in V^*. \tag{210}$$

Comme son application lors de la phase reformulation de la formulation F1-RLT, l'inégalité valide $x_{0j} \leq y_j, \forall j \in \delta^+(0)$, implique la contrainte (211) qui suit

$$(x_{0j} \leq y_j) * s_j, \forall j \in \delta^+(0). \tag{211}$$

Ensuite, on utilise l'inégalité valide (81) basée sur les flots pour obtenir la contrainte non-linéaire suivante:

$$\left(\sum_{i \in \delta^-(j)} x_{ij} \geq x_{jk} \right) * s_j, \ \forall j \in V^*, \ \forall (j,k) \in A, \tag{212}$$

Enfin, on part de la contrainte (201) pour développer les contraintes qui suivent

$$(0 \leq s_j \leq (n - \vartheta_j)y_j) * y_j, \ \forall j \in V^*, \tag{213}$$

$$(0 \leq s_j \leq (n - \vartheta_j)y_j) * x_{ij}, \ \forall (i,j) \in A, \ j \in V^*, \tag{214}$$

$$(0 \leq s_i \leq (n - \vartheta_i)y_i) * x_{ij}, \ \forall (i,j) \in A, \ i \in V^*, \tag{215}$$

Dans le paragraphe suivant, nous explicitons la deuxième phase de la technique RLT.

Phase de linéarisation Afin de linéariser les termes non-linéaires obtenus lors de la phase de reformulation, on applique les substitutions

$$w_{ij} = s_j x_{ij}, \ \forall (i,j) \in A, \tag{216}$$

$$d_{ij} = s_i x_{ij}, \ \forall (i,j) \in A, \tag{217}$$

$$b_j = s_j y_j, \ \forall j \in V^*. \tag{218}$$

Ainsi,On obtient la formulation F3-SC-RLT du PCSTP qui suit:

$$\textbf{F3-SC-RLT} : \ \text{Minimiser} \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{j \in V^*} \gamma_j(1 - y_j)$$

sous contraintes (118)-(120), (122)-(123), (196)-(198), (202),

$$w_{ij} \leq d_{ij} - x_{ij}, \ \forall (i,j) \in A, \tag{219}$$

$$\sum_{k\in\delta^+(i)} w_{ik} + \sum_{k\in\delta^+(i)} x_{ik} = \sum_{k\in\delta^+(i)} f_{ik}, \ \forall i \in V^*, \tag{220}$$

$$\sum_{j\in\delta^+(0)} w_{0j} + \sum_{j\in\delta^+(0)} x_{0j} = \sum_{j\in V^*} y_j, \tag{221}$$

$$w_{ij} + d_{ji} \leq b_j, \ \forall j \in V^*, \ \forall (i,j),(j,i) \in A, \tag{222}$$

$$\sum_{i\in\delta^-(j)} w_{ij} = b_j, \ \forall j \in V^*, \tag{223}$$

$$w_{0j} \leq b_j, \ \forall j \in \delta^+(0), \tag{224}$$

$$\sum_{i\in\delta^-(j)} w_{ij} \geq d_{jk}, \ \forall j \in V^*, \ (j,k) \in A, \tag{225}$$

$$0 \leq b_j \leq (n - \vartheta_j - 1)y_j, \ \forall j \in V^*, \tag{226}$$

$$0 \leq w_{ij} \leq (n - \vartheta_j - 1)y_j, \ \forall j \in V^*, \ (i,j) \in A, \tag{227}$$

$$0 \leq d_{ij} \leq (n - \vartheta_i - 1)y_i, \ \forall i \in V^*, \ (i,j) \in A \tag{228}$$

Les contraintes (219)-(228) sont obtenues en linéarisant respectivement les contraintes (206)-(215) en appliquant les relations (216)-(218). En plus, pour la formulation (220), on remplace chaque variable s_i, $i \in V^*$, par son expression (200) en fonction des f_{ik}, $(i,k) \in A$. Pour la contrainte (226), on utilise le fait que les variables y_j, $j \in V^*$, sont binaires, i.e $y_j^2 = y_j$. Enfin, ces inégalités valides pour tout arc $(i,j) \in A$, $j \in V^*$, $x_{ij}y_j \leq y_j$ et $x_{ij}y_i \leq y_i$ sont aussi appliquées pour obtenir respectivement les contraintes (227) et (228).

Ainsi, nous avons développé douze formulations compactes partitionnées en trois familles. Dans la section qui suit, nous comparons théoriquement les relaxations linéaires de ces modèles.

6.5 Comparaison des relaxations linéaires des formulations

Dans ce paragraphe, nous présentons une étude théorique des relations de dominance entre les relaxations linéaires des différentes formulations développées pour le PCSTP. Une première approche de comparaison des formulations compactes est de spécifier leurs complexités respectives présentées dans le tableau 7.1 qui suit.

Formulation	nombre de variables	nombre des contraintes	Type de formulation
$F1-MTZ$	$O(n^2)$	$O(n^2)$	1
$F1-DL$	$O(n^2)$	$O(n^2)$	1
$F1-RLT$	$O(n^2)$	$O(n^2)$	1
$F2-P$	$O(n^2)$	$O(n^3)$	2
$F2-PL0$	$O(n^2)$	$O(n^3)$	2
$F2-PL1$	$O(n^2)$	$O(n^3)$	2
$F2-PL2$	$O(n^2)$	$O(n^3)$	2
$F2-RLT$	$O(n^3)$	$O(n^3)$	3
$F3-MC$	$O(n^3)$	$O(n^3)$	3
$F3-SC$	$O(n^2)$	$O(n^2)$	1
$F3-SC-MTZ$	$O(n^2)$	$O(n^2)$	1
$F3-SC-RLT$	$O(n^2)$	$O(n^2)$	1

Tableau 6.1: Complexité des formulations compactes

Avant de passer à la comparaison théorique des relaxations linéaires des formulations compactes proposées, nous commençons par présenter une formulation exponentielle, en terme de nombre de contraintes, qu'on note la formulation **DMST** (Directed Minimum Spanning Tree Formulation) définie par

$$\textbf{DMST}: \quad \text{Minimiser} \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{j\in V^*} \gamma_j(1-y_j)$$

sous contraintes: (119)-(120), (123),

$$\sum_{(i,j)\in E(S)} x_{ij} \leq \sum_{i\in S\setminus\{k\}} y_i, \qquad \forall S \subseteq V,\ |S| \geq 2,\ \forall k \in S, \tag{229}$$

$$x_{ij} \in [0,1], \qquad \forall (i,j) \in A. \tag{230}$$

Où pour tout sous-ensemble $S \subset V$ avec $|S| \geq 2$, $E(S) = \{(i,j) \in A : i \in S, j \in S\}$.

La contrainte (229) domine la contrainte (126) classique d'élimination de sous-tours pour les problèmes d'arbre introduite depuis 1954 par Dantzig et al.[43]. En fixant à 1 quelques variables binaires du vecteur $\mathbf{y}$, le problème revient à résoudre le problème d'arbre de poids minimal couvrant le sous-ensemble des noeuds V' tel que $y_j = 1$, $j \in V'$. Grâce à la propriété d'intégralité du problème d'arbre couvrant de poids minimal, il suffit de relaxer la contrainte d'intégrité (122) des variables $\mathbf{x}$. Ainsi, on obtient la contrainte (230).

Notons qu'ensemble les contraintes (120) et (229) sont équivalentes à la contrainte (121), i.e elles garantissent que la solution du PCSTP soit une arborescence. En particulier, tout arborescence admet nécéssairement le noeud 0 comme racine. En effet, comme il n'y a pas d'arcs entrants au noeud racine 0 qui n'admet ni profit ni pénalité, et comme tout sommet j couvert par l'arborescence a exactement 1 seul arc entrant (120), et en plus il n'y a pas de sous-tours (229), alors il existe nécéssairement un sommet et un seul qui admet le noeud 0 comme prédécésseur.

Nous rappelons que nous avons déjà considéré une formulation exponentielle **DCF** du paragraphe 6.2.2 que nous rappelons

$$\textbf{DCF}: \quad \text{Minimiser} \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{j\in V^*} \gamma_j(1-y_j)$$

sous contraintes: (119), (122)-(123),

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq y_k, \quad \forall S \subset V,\ 0 \in S,\ k \in V \setminus S. \tag{231}$$

Où, pour tout sous-ensemble non vide $S \subset V$, $\delta^+(S) = \{(i,j) \in A : i \in S \text{ et } j \in V \setminus S\}$.

Nous rappelons aussi que toute solution optimale de la formulation DCF vérifie nécéssairement la contrainte (120). Et par suite, toute solution optimale de la relaxation linéaire de DCF vérifie nécessairement l'inégalité (232) qui suit

$$\sum_{(i,j)\in\delta^-(S)} x_{ij} \geq \sum_{i\in\delta^-(k)} x_{ik}, \qquad \forall S \subset V \text{ tel que } 0 \in V \setminus S,\ \forall k \in S. \tag{232}$$

Notons qu'en écrivant la contrainte (232) dans le cas particulier $S = \{0\}$, on déduit que la racine est nécessairement couverte par l'arborescence i.e on obtient la contrainte (118).

Après avoir exposé les formulations exponnentielles DMST et DCF, nous passons à la comparaison théorique des relaxations linéaires des différentes formulations proposées pour le PCSTP. Notons $Z_{LP}(F)$ la valeur optimale de la relaxation linéaire de la formulation F du PCSTP.

Proposition 29 *$Z_{LP}(DMST) \leq Z_{LP}(DCF)$.*

Preuve. Considérons ($\mathbf{x}$,$\mathbf{y}$) une solution optimale de la relaxation linéaire de la formulation DCF, montrons qu'elle vérifie toutes les contraintes de la formulation DMST. Il suffit de montrer que ($\mathbf{x}$,$\mathbf{y}$) vérifie la contrainte (229).

Soit un sous-ensemble $S \subseteq V$ avec $|S| \geq 2$. Deux cas peuvent se présenter.

Premier cas: le noeud racine $0 \in S$.

Dans ce cas, l'inégalité suivante est vérifiée:
$\sum_{(i,j)\in E(S)} x_{i,j} \leq \sum_{i\in S} \sum_{j\in\delta^-(i)} x_{ji} = \sum_{i\in S, i\neq 0} \sum_{j\in\delta^-(i)} x_{ji}$ car il n'existe pas d'arcs entrants noeud 0. Ainsi, on obtient l'inégalité $\sum_{(i,j)\in E(S)} x_{i,j} \leq \sum_{i\in S, i\neq 0} y_i = \sum_{i\in S} y_i - 1$, car $y_0 = 1$. Or, $\forall k \in S$, $y_k \leq 1$. Donc, on a
$\sum_{i\in S} y_i - 1 = \sum_{i\in S\setminus\{k\}} y_i + (y_k - 1) \leq \sum_{i\in S\setminus\{k\}} y_i$, $k \in S$. Ainsi, la contrainte (229) est vérifiée.

Deuxième cas: le noeud racine $0 \notin S$.
On a l'égalité $\sum_{(i,j)\in E(S)} x_{i,j} = \sum_{i\in S} \sum_{j\in\delta^-(i)} x_{ji} - \sum_{(k,l)\in\delta^-(S)} x_{kl}$.
Or, d'après la contrainte (120) on a: $\forall i \in S$, $\sum_{j\in\delta^-(i)} x_{ji} = y_i$. Donc, on obtient $\sum_{(i,j)\in E(S)} x_{i,j} = \sum_{i\in S} y_i - \sum_{(k,l)\in\delta^-(S)} x_{kl} = \sum_{i\in S\setminus\{k\}} y_i + y_k - \sum_{(k,l)\in\delta^-(S)} x_{kl}, \forall k \in S$. En plus, la contrainte (231) s'écrit: $\sum_{(k,l)\in\delta^-(S)} x_{kl} \geq y_k, \forall k \in S$. Ainsi, on peut conclure que la contrainte (229) est vérifiée. ■

Une conséquence immédiate de la proposition 38 est le fait que la formulation DCF ne peut pas être renforcée en lui ajoutant des contraintes de la formulation DMST. En particulier, il est inutile d'ajouter la contrainte valide (120) afin de renforcer la formulation DCF telque nous l'avons déjà présenté dans le chapitre (5). Or, cette contrainte (120) est vérifiée par toute solution *optimale* du DCF. Ce qui fait, elle reste très utile dans le cadre de l'algorithme de génération de contraintes telle que nous l'avons appliqué.

Proposition 30 *$Z_{LP}(DCF) = Z_{LP}(F3\text{-}MC)$.*

Preuve. *La preuve peut être considérée immédiate puisqu'elle découle du théorème du flot max-coupe min.*

Pour la clarté de ce document, on peut détailler la preuve comme suit:
1- Montrons que Z_{LP}(F3-MC) $\geq Z_{LP}$(DCF).

Considérons (**x,y,f**) une solution réalisable de la relaxation linéaire de la formulation F3-MC. Soit $k \in V^*$, considérons le problème du flot circulant du sommet 0 vers le sommet k. Il est possible de faire circuler une quantité y_k du flot de commodité k du sommet 0 vers le sommet k à travers le graphe G, où pour tout arc $(i,j) \in A$, x_{ij} est la capacité de l'arc (i,j) pour tout flot de commodité k, $k \in V^*$.

Pour tout sommet $k \in V^*$, le théorème du flot max-coupe min garantit que toute coupe séparant les sommets 0 et k admet une capacité au moins égale à y_k. Ce qui se traduit par la contrainte (231). On déduit que (**x,y**) est bien une solution réalisable pour la relaxation linéaire de la formulation DCF.

2- Montrons que Z_{LP}(F3-MC) $\leq Z_{LP}$(DCF)

Considérons (**x,y**) une solution réalisable de la relaxation linéaire de la formulation DCF. Considérons un sommet $k \in V^*$. Pour tout arc $(i,j) \in A$,

on peut lui associer la valeur x_{ij} comme capacité pour le flot de commodité k. Or, toute solution optimale de DCF vérifie la contrainte (120) qui peut s'écrire de la forme $y_k = \sum_{i \in \delta^-(k)} x_{ik} = \sum_{(i,j) \in \delta^+(V \setminus \{k\})} x_{ij}, \forall k \in V^*$.
Ainsi, la contrainte (231) s'écrit:
$\sum_{(i,j) \in \delta^+(S)} x_{ij} \geq \sum_{(i,j) \in \delta^+(V \setminus \{k\})} x_{ij}, \ \forall S \subset V, \ 1 \in S, \ k \in V \setminus S$, Il s'agit bien d'une coupe minimale $C = (V \setminus \{k\} : k)$. D'après le théorème du flot max-coupe min, on peut construire un flot de valeur égale à y_k qui circule entre les sommets 0 et k. Ainsi, on peut construire une solution réalisable (**x,y,f**) de la relaxation linéaire de la formulation F3-MC.

On peut conclure que l'ensemble des solutions réalisable de la formulation DCF est la projection de celui de la formulation F3-MC sur l'espace des variables (**x,y**). ■

Proposition 31 *Z_{LP}(F1-MTZ') $\leq Z_{LP}$(DMST)*

Preuve. Cette preuve est largement inspiré de celle de Polzin et Daneshmand [171] pour le problème de Stiener dans les graphes. Mais, pour la complétude de ce document, nous exposons cette preuve qui est effectuée avec F1-MTZ' qui représente une simplification de la formulation F1-MTZ. En effet, on considère la formulation MTZ classique, où on remplace dans F1-MTZ la contrainte (127) par la contrainte (233) suivante:

$$(n+1)x_{ij} + u_i - u_j \leq n, \qquad \forall (i,j) \in A, \tag{233}$$

On peut ne pas avoir de contraintes sur les bornes des variables **u**. Mais, pour donner un sens à ces variables, on ajoute la contrainte simple

$$0 \leq u_j \leq n, \quad \forall j \in V. \tag{234}$$

Ainsi, on a la formulation basique F1-MTZ' définit par

$$\textbf{F1-MTZ':} \ \text{Minimiser} \left\{ \begin{array}{c} \sum_{(i,j) \in A} c_{ij}x_{ij} + \sum_{j \in V^*} \gamma_j(1-y_j) : (118)\text{-}(120), \\ (122)\text{-}(123) \text{ et } (233)\text{-}(234) \end{array} \right\}.$$

Considérons (**x**,**y**) une solution réalisable (optimale) de la relaxation linéaire de la formulation DMST, montrons qu'on peut construire un vecteur **u** qui vérifie les contraintes (233)-(234). On obtient ainsi (**x**,**y**,**u**) une solution réalisable la relaxation linéaire de la formulation F1-MTZ'.

On part d'un vecteur $\mathbf{u}$ quelconque. On peut prendre par exemple $u_i = 0$, $\forall i \in V$. Pour chaque arc $(i,j) \in A$, on définit le réel suivant:

$$s_{ij} = n - (u_i - u_j + (n+1)x_{ij}) \tag{235}$$

Un arc $(i,j) \in A$ est appelée (bon) si $s_{ij} \geq 0$, (normal) si $s_{ij} \leq 0$, et (mauvais) si $s_{ij} < 0$.
Si aucun arc $(i,j) \in A$ n'est (mauvais) alors la contrainte (233) est vérifié. Sinon, considérons un arc $(i,j) \in A$ qui est (mauvais).
On peut montrer comment peut-on augmenter la valeur de u_j afin que l'arc (i,j) devient (bon) sans qu'aucun (bon) arc ne devienne (mauvais). D'autres noeuds $p \in V$ peuvent éventuellemnt voir leur u_p augmenter. Ainsi, il suffit de répéter cette procédure jusqu'à ce que tous les arcs deviennent (bon):

Etape 0: considérons un arc $(i,j) \in A$ de type (mauvais).
A chaque étape, on note W_j l'ensemble des noeuds $k \in V$ qu'on peut atteindre à partir du noeud j avec des chemins composés uniquement d'arcs de type (normal).
On définit le paramètre $\Delta = \min\left\{s_{kl} : (k,l) \in \delta^+(W_j)\right\}$ si l'ensemble W_j est non vide, $= \infty$ sinon.
Pour augmenter s_{ij} il suffit d'augmenter u_j.

Etape 1: on augmente pour tout $p \in W_j$, $u_p := u_p + \min\{-s_{ij}, \Delta\}$.
Ainsi, on constate que :

- aucun arc $(k,l) \in \delta^+(W_j)$ ne devient (mauvais),
- pour tous les arcs $(p,q) \in \delta^-(W_j)$, s_{pq} ne diminue pas,
- pour tous les arcs $(p,q) \in A$ tel que $p \in W_j$ et $q \in W_j$ ou $p \notin W_j$ et $q \notin W_j$, s_{pq} reste constant.

Etape 2: Test: si tous les arcs ne sont pas (mauvais) alors FIN, sinon aller à Etape 0.

Grâce à cette procédure, la valeur de u_j augmente à chaque itération. Or, une seule situation peut empêcher l'arc (i,j) de devenir (bon). Elle est la suivante: à une itération, le noeud i peut être absorbé par l'ensemble W_j ,i.e $i \in W_j$. En d'autres termes, u_i et u_j augmentent et s_{ij} ne change pas.
Dans ce cas, par définition même de l'ensemble W_j, il existe un chemin de j vers i avec seulement des arcs de type (normal). Ainsi, il existe un cycle $C := \{i, j = n_1, ..., n_K = i\}$ telque $s_{n_K n_1} < 0$ et $s_{n_{t-1} n_t} \leq 0, \forall t = 2, ..., K$ avec $2 \leq K \leq n$. En utilisant l'égalité (235) et en sommant ces inégalités sur tous

les arcs du cycle C, on obtient l'inégalité suivante:

$$nK - (n+1) \sum_{(i,j)\in C} x_{ij} < 0$$

Et par suite, on a $\sum_{(i,j)\in C} x_{ij} > K\frac{n}{n+1}$.

D'autre part, comme (**x,y**) est une solution réalisable (optimale) de la DMST, alors elle vérifie la contrainte (126) et par suite, on a $\sum_{(i,j)\in C} x_{ij} \leq K - 1$. Ainsi, on obtient les implications qui suivent $\frac{n-1}{n} < \frac{K-1}{K} \Rightarrow 1 - \frac{1}{n} < 1 - \frac{1}{K}$ $\Rightarrow \frac{1}{n} > \frac{1}{K} \Rightarrow n < K$. Ce qui est absurde. ■

Proposition 32 *Z_{LP}(F1-MTZ') $\leq Z_{LP}$(F3-MC). En plus, il existe des instances telles que Z_{LP}(F1-MTZ) $< Z_{LP}$(F3-MC).*

Preuve. Par transitivité des relations de dominance, on a
Z_{LP}(F1-MTZ') $\leq Z_{LP}$(DMST) $\leq Z_{LP}$(DCF) $= Z_{LP}$(F3-MC).
Pour l'instance benchmark B02 de la bibliothéque *SteinLib* [131] (voir détail au paragraphe 5.7), on a trouvé que:
Z_{LP}(F1-MTZ) $= 102,83 < Z_{LP}$(F3-MC) $= 153$. ■

Proposition 33 *Z_{LP}(F3-SC') $\leq Z_{LP}$(F3-MC). En plus, il existe des instances telles que Z_{LP}(F3-SC) $< Z_{LP}$(F3-MC).*

Preuve. La formulation F3-SC' correspond à la formulation F3-SC sans la contrainte valide (202). Considérons une solution (**x,y,f**) réalisable de la relaxation linéaire de F3-MC. Donc, elle vérifie les contraintes (192)-(195). Il suffit de montrer qu'elle vérifie les contraintes (196)-(197). On se base sur l'application de la relation suivante: $f_{ij} = \sum_{k\in V^*} f_{ij}^k$, $\forall (i,j) \in A$.
En faisant la somme des contraintes (192) sur tous les $k \in V^*$, on obtient l'égalité: $\sum_{k\in V^*} \sum_{j\in\delta^+(1)} f_{1j}^k = \sum_{k\in V^*} y_k \Leftrightarrow \sum_{j\in\delta^+(1)} f_{1j} = \sum_{k\in V^*} y_k$. Ce qui correspond à la contrainte (196).
En sommant des contraintes (194) sur tous les $k \in V^*$, on obtient:
$\sum_{i\in\delta^-(j)} \sum_{k\in V^*, j\neq k} f_{ij}^k - \sum_{i\in\delta^+(j)} \sum_{k\in V^*, j\neq k} f_{ji}^k = 0, \forall j \in V^*$.
$\Rightarrow \sum_{i\in\delta^-(j)} \left(f_{ij} - f_{ij}^j\right) - \sum_{i\in\delta^+(j)} \left(f_{ji} - f_{ji}^j\right) = 0, \forall j \in V^*$.

$\Rightarrow \sum_{i\in\delta^-(j)} f_{ij} - y_j - \sum_{i\in\delta^+(j)} f_{ji} = 0, \forall j \in V^*$, car la contrainte (193) est vérifiée et en plus par construction, on a $f^j_{ji} = 0, \forall i, j \in V^*, i \neq j$. Ainsi, on obtient la contrainte (197).
En faisant la somme des contraintes (195) sur tous les $k \in V^*$, on obtient: $0 \leq \sum_{k\in V^*} f^k_{ij} \leq \sum_{k\in V^*} x_{ij} \Rightarrow 0 \leq f_{ij} \leq (n-1)x_{ij}$.
Comme pour tout arc $(i,j) \in A$, f_{ij} représente la quantité de flot circulant sur l'arc (i,j), alors, les relations suivantes sont valides: si $f_{ij} = 0$, alors $x_{ij} = 0$ et si $x_{ij} = 1$ alors $f_{ij} \geq 1$. Par suite, l'inégalité suivante est valide: $x_{ij} \leq f_{ij}, \forall (i,j) \in A$. D'autre part, il est clair que par construction, pour tout arc $(i,j) \in A$, la quantité f_{ij} ne peut pas être strictement supérieure à $n - \vartheta_i$. Et parsuite, on peut conclure que $f_{ij} \leq (n - \vartheta_i)x_{ij}, \forall (i,j) \in A$. Ainsi, on déduit la contrainte (198).

Pour l'instance B02, on a trouvé que:
Z_{LP}(F3-SC) $= 110,57 <$ Z_{LP}(F3-MC) $= 153$. ■

Proposition 34 *Z_{LP}(F1-DL) $\leq$ Z_{LP}(F2-P1). En plus, il existe des instances telles que Z_{LP}(F1-DL) $<$ Z_{LP}(F2-P1).*

Preuve. Partant d'une solution réalisable (**x,y,z**) de la relaxation linéaire de F2-P, montrons qu'on peut construire une solution (**x,y,u**) réalisable pour la relaxation linéaire de F1-DL. Il suffit de définir les composantes du vecteur **u** comme suit: $u_j = \sum_{i\in V, i\neq j} z_{ij}$, $\forall j \in V^*$ avec $u_0 = 0$. Ainsi, les contraintes (167) et (168) induisent respectivement les contraintes (132) et (131).
Pour le problème B02, on a trouvé que:
Z_{LP}(F1-DL) $= 131,50 <$ Z_{LP}(F2-P) $= 142,71$. ■

Proposition 35 *Z_{LP}(F1-DL') $\leq$ Z_{LP}(F1-RLT). En plus, il existe des instances telles que Z_{LP}(F1-DL) $<$ Z_{LP}(F1-RLT).*

Preuve. La formulation F1-DL' est la contrainte F1-DL avec la contrainte (128) à la place de la contrainte (132). Soiy (**x**,**y**,**t**,**v**) une solution réalisable de la relaxation linéaire du problème F1-RLT, montrons que (**x**,**y**,**v**) est une solution réalisable de la relaxation linéaire du problème F1-DL'. Il suffit de vérifier la contrainte (131) car les contraintes (150) et (128)

sont identiques. La contrainte (151) est équivalente aux deux inégalités suivantes:

$$v_j - ny_j + (n-1)x_{ij} + nx_{ji} \leq t_{ij} + t_{ji},\ \forall (i,j),(j,i) \in A, \tag{236}$$

$$t_{ij} + t_{ji} \leq v_j - \vartheta_j y_j + (\vartheta_j - 1)x_{ij} + \vartheta_j x_{ji},\ \forall (i,j),(j,i) \in A. \tag{237}$$

Partant de l'inégalité (236), on interchange les indices i et j et on multiplie par (-1) l'inégalité, on obtient:

$$-t_{ji} - t_{ij} \leq -v_i + ny_i - (n-1)x_{ji} - nx_{ij},\ \forall (i,j),(j,i) \in A. \tag{238}$$

Par suite, il suffit de sommer les inégalités (237) et (238) pour obtenir la contrainte (131). Enfin, on ajoute $v_1 = 0$ pour conclure que ($\mathbf{x}$,$\mathbf{y}$,$\mathbf{v}$) est une solution réalisable de F1-DL'.
Pour l'instance B02, on a trouvé que:
Z_{LP}(F1-DL) $= 131,50 < Z_{LP}$(F1-RLT) $= 142,71$. ■

Proposition 36 *Z_{LP}(F2-P) $\leq$ Z_{LP}(F2-PL0) $\leq$ Z_{LP}(F2-PL2). En plus, il existe des instances telles que Z_{LP}(F2-P) $<$ Z_{LP}(F2-PL0) $<$ Z_{LP}(F2-PL2).*

Preuve. La preuve est évidente. Pour l'instance B02, on a trouvé que:
Z_{LP}(F2-P)=$142,71 < Z_{LP}$(F2-PL0)=$142,74 < Z_{LP}$(F2-PL2)=$142,95$. ■

Proposition 37 *Z_{LP}(F2-P) $\leq$ Z_{LP}(F2-PL1) $\leq$ Z_{LP}(F2-PL2). En plus, il existe des instances telles que Z_{LP}(F2-P) $<$ Z_{LP}(F2-PL1) $<$ Z_{LP}(F2-PL2).*

Preuve. La preuve est évidente. Pour l'instance B02, on a trouvé que:
Z_{LP}(F2-P)=$142,71 < Z_{LP}$(F2-PL1)=$142,93 < Z_{LP}$(F2-PL2)=$142,95$. ■

Proposition 38 *Z_{LP}(F1-MTZ) $\leq$ Z_{LP}(F1-DL). En plus, il existe des instances telles que Z_{LP}(F1-MTZ) $<$ Z_{LP}(F1-DL).*

Preuve. La preuve est évidente. Pour l'instance B02, on a trouvé que:
Z_{LP}(F1-MTZ) $= 102,83 < Z_{LP}$(F1-DL) $= 131,50$. ■

Proposition 39 *Z_{LP}(F3-SC) $\leq$ Z_{LP}(F3-SC-RLT). En plus, il existe des instances telles que Z_{LP}(F3-SC) $<$ Z_{LP}(F3-SC-RLT).*

Preuve. La preuve est évidente. Pour l'instance B02, on a trouvé que:
Z_{LP}(F3-SC) $= 110,57 < Z_{LP}$(F3-SC-RLT) $= 113,91$. ■

Proposition 40 $Z_{LP}(F2\text{-}P) \leq Z_{LP}(F2\text{-}RLT)$.

Preuve. La preuve est évidente. ■

Proposition 41 $Z_{LP}(F3\text{-}SC) \leq Z_{LP}(F3\text{-}SC\text{-}MTZ)$.

Preuve. La preuve est évidente. ■

Un schéma récapitulatif des relations de dominance des relaxations linéaires des formulations discutées dans ce chapitre est présentée dans la figure 7.2. Une flèche de la formulation F_1 vers la formulation F_2 indique que la relaxation linéaire de la formulation F_2 domine (plus forte que) celle de F_1. Notons que la relation de "dominance stricte" est transitive. Les formulations de la même boite sont équivalentes en terme de relaxation linéaire.

Figure 6.2: Hiérarchie des relaxations linéaires des formulations du PCSTP

Ainsi, nous avons présenté une comparaison théorique des relaxations linéaires des formulations polynomiales que nous avons développées pour le PCSTP. Dans la suite, nous présentons une comparaison empirique de leur performance en terme de relaxation linéaire, de résolution approchée et aussi de résolution exacte du PCSTP.

6.6 Etude expérimentale

Le solveur de programmes linéaires utilisé est CPLEX. Il existe une raison majeure à l'utilisation de CPLEX plutôt qu'un autre solveur. En effet, CPLEX est déjà éprouvé et il est devenu une référence dans le domaine de la résolution de programmes linéaires, de telle sorte qu'en pratique quasiment aucun problème liè au solveur ne se présente. Nous avons appliqué ce solveur sous sa dernière version disponible sur le marché, à savoir la version 11.

Similaire à l'heuristique HSPI et la procédure branch-and-cut, nous présentons une illustration des résultats numériques sur 53 instances Benchmark réparties en 4 classes de la bibiliothéque *SteinLib* [131]. Il s'agit en partie des mêmes instances décrites dans le paragraphe 4.7. En notant $n \equiv$ nombre de noeuds et $m \equiv$ nombre d'arcs, la répartition des instances est la suivante:

- Toute la classe B qui compte 18 instances dont la taille varie de $n = 50$, $m = 63$ à $n = 100$, $m = 200$,
- Toute la classe C qui compte 20 instances dont la taille varie de $n = 500$, $m = 625$ à $n = 500$, $m = 12500$,
- 10 instances de la classe D dont la taille varie de $n = 1000$, $m = 1250$ à $n = 1000$, $m = 2000$,
- 5 instances de la classe E dont la taille varie de $n = 2500$, $m = 3125$.

Autre que les coûts des arêtes, les paramètres du problème sont générés aléatoirement comme suit:

- les profits $(p_i)_{i \in V^*}$ sont des entiers générés aléatoirement entre 1 et 10.
- les pénalités $(\gamma_i)_{i \in V^*}$ sont calculées selon la formule $\gamma_i = \lceil 0.5 p_i \rceil$, $i \in V^*$.
- le quota minimum à collecter est égal à $Q = \left\lceil \alpha \sum_{j \in V^*} p_j \right\rceil$ avec $\alpha = 0.5$.

Nous rappelons qu'en prenant $\alpha = 0.5$, nous supposons qu'on cherche à connecter de l'ordre de 50% des sommets du réseau.

6.6.1 Comparaison empirique des relaxations linéaires

Dans ce paragraphe, on se propose de tester la performance des formulations compactes développées en terme de relaxation linéaire. D'abord, nous avons commencé par résoudre la relaxation linéaire de toutes les formulations compactes dévéloppées aux 18 instances de la classe B. Les détails des résultats ainsi que les temps de résolution sont disponibles dans le tableau 6.2. La signification des colonnes est la suivante:

- $Inst. \equiv$ le nom de l'instance,
- $Gap = 100 \times \frac{Z^* - LB}{Z^*}$, où LB est la borne inférieure solution de la relaxation linéaire et Z^* est la solution optimale,
- $Temps \equiv$ le temps CPU nécéssaire au calcul de LB en secondes.

	F1-MTZ		F1-DL		F1-RLT		F3-SC		F3-SC-MTZ		F3-SC-RLT		F3-MC		F2-P		F2-PL0		F2-PL1		F2-PL2		F2-RLT	
Inst.	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps
B.01	23.66	0.03	6.93	0.02	0.61	0.02	20.92	0.03	20.92	0.05	15.39	0.13	0.48	0.22	0.62	1206.20	0.62	1128.38	0.62	965.09	0.62	1183.13	0.62	463.33
B.02	32.79	0.00	14.05	0.02	6.72	0.03	27.73	0.03	27.73	0.03	25.55	0.16	**0.00**	0.48	6.72	1335.67	6.71	1039.02	6.58	1266.13	6.57	1078.66	6.72	391.02
B.03	18.45	0.00	8.35	0.00	3.65	0.03	14.91	0.05	14.91	0.03	14.46	0.14	0.17	0.13	3.65	1529.92	3.65	1099.95	3.31	656.59	3.31	622.33	3.65	568.06
B.04	17.09	0.02	7.11	0.00	4.53	0.05	13.85	0.05	13.85	0.06	13.82	0.50	**0.00**	1.05	4.58	768.19	4.58	898.83	4.37	718.05	4.34	586.67	4.58	512.66
B.05	15.43	0.00	3.59	0.00	0.47	0.03	14.52	0.05	14.52	0.06	13.68	0.34	0.47	1.08	0.47	1004.11	0.47	1175.24	0.47	725.42	0.47	1224.31	0.47	466.39
B.06	25.13	0.00	9.09	0.02	0.83	0.05	21.96	0.06	21.96	0.08	21.96	0.53	**0.00**	4.09	0.83	2291.33	0.83	937.24	0.83	589.83	0.83	1991.06	0.83	2557.89
B.07	25.09	0.02	10.09	0.00	0.93	0.06	22.57	0.09	22.57	0.08	20.17	0.45	0.45	3.42	-	-	-	-	-	-	-	-	-	-
B.08	18.41	0.00	5.10	0.02	1.39	0.06	16.43	0.08	16.43	0.08	14.88	0.45	0.42	2.02	-	-	-	-	-	-	-	-	-	-
B.09	22.23	0.02	7.49	0.00	2.78	0.05	19.39	0.08	19.39	0.09	18.93	0.42	0.37	1.33	-	-	-	-	-	-	-	-	-	-
B.10	16.02	0.00	4.58	0.02	0.15	0.09	14.75	0.14	14.75	0.17	14.74	1.86	**0.00**	5.64	-	-	-	-	-	-	-	-	-	-
B.11	18.37	0.02	6.83	0.00	1.64	0.13	16.04	0.14	16.04	0.17	16.03	1.22	**0.00**	8.02	-	-	-	-	-	-	-	-	-	-
B.12	21.80	0.00	4.84	0.02	0.00	0.11	20.05	0.16	20.05	0.16	20.03	1.50	**0.00**	21.88	-	-	-	-	-	-	-	-	-	-
B.13	25.45	0.00	11.45	0.02	0.98	0.09	23.51	0.16	23.51	0.17	21.47	0.80	0.33	6.20	-	-	-	-	-	-	-	-	-	-
B.14	21.71	0.02	7.80	0.02	2.10	0.09	19.71	0.16	19.71	0.16	18.28	0.88	0.29	4.28	-	-	-	-	-	-	-	-	-	-
B.15	29.07	0.02	7.03	0.02	0.90	0.09	26.68	0.16	26.68	0.16	26.56	0.94	0.21	14.05	-	-	-	-	-	-	-	-	-	-
B.16	21.29	0.02	8.65	0.02	0.00	0.27	19.53	0.28	19.53	0.31	17.51	4.72	**0.00**	95.55	-	-	-	-	-	-	-	-	-	-
B.17	15.01	0.02	4.28	0.02	0.00	0.14	14.69	0.27	14.69	0.28	14.65	6.66	**0.00**	15.09	-	-	-	-	-	-	-	-	-	-
B.18	18.24	0.02	8.70	0.02	1.79	0.22	16.54	0.27	16.54	0.30	16.08	4.61	**0.00**	75.72	-	-	-	-	-	-	-	-	-	-
Moyenne	*21.40*	*0.01*	*7.55*	*0.01*	*1.64*	*0.09*	*19.10*	*0.12*	*19.10*	*0.14*	*18.01*	*1.46*	*0.18*	*14.46*	*2.81*	*1355.90*	*2.81*	*1046.44*	*2.70*	*820.19*	*2.69*	*1114.36*	*2.81*	*826.56*
Ecart type	*4.85*		*2.65*		*1.79*		*4.23*		*4.23*		*3.93*		*0.20*		*2.58*		*2.58*		*2.49*		*2.48*		*2.58*	

Tableau 6.2: Performance des relaxations linéaires des formulations compactes sur la classe B

Le tableau récapitulatif des résultats des relaxations linéaires des formulations est le suivant:

Formulation	**Gap moyen**	**Gap maximal**	**Ecart Type**	**Temps Moyen**
F1-MTZ	21.40	32.79	4.85	0.01
F1-DL	7.55	14.05	2.65	0.01
F1-RLT	*1.64*	*6.72*	*1.79*	*0.09*
F2-P	2.81	6.72	2.58	1 355.90
F2-PL0	2.81	6.71	2.58	1 046.440
F2-PL1	2.70	6.58	2.49	820.185
F2-PL2	2.69	6.57	2.48	1 114.359
F2-RLT	2.81	6.72	2.58	826.558
F3-MC	*0.18*	*0.48*	*0.20*	*14.45*
F3-SC	19.10	27.73	4.23	0.12
F3-SC-MTZ	19.10	27.73	4.23	0.13
F3-SC-RLT	18.01	26.56	3.93	1.46

Tableau 6.3: Tableau récapitulatif des résultats des relaxations linéaires des formulations compactes pour la classe B

A partir de ces tableaux de résultats, on peut conclure que:

- Toutes les formulations de la famille F2 sont très limitées en terme de taille des instances. Même si pour les six instances B01-B06 qu'elle peuvent résoudre, les gaps sont pratiquement identitiques à ceux de F1-RLT, les temps de cacul sont extrêment importants.
- Contrairement à la famille de formulation F1, le fait d'appliquer et de développer la technique RLT n'a pas amélioré la performance de F2. En effet, les relaxations linéaires des formulations F2-RLT et F2-P donnent les mêmes résultats. Par contre, on observe une diminution du temps de calcul due à l'application de la technique RLT.
- Les formulations F3-SC-MTZ et F3-SC ont la même performance en terme de relaxation linéaire.

- Pour ce qui concerne la famille de formulations F1, on remarque l'effet du lifting des contraintes MTZ. En effet, le gap moyen passe de l'ordre de 21.5% pour F1-MTZ à 7.5% pour F1-DL. En plus, toutes les formulation de F1 sont extrêment rapides.

- Les meilleures performances sont celles des formulations F3-MC et F1-RLT. Plus précisement, la borne de la relaxation linéaire de F3-MC est égale à l'optimum pour 50% des instances contre 16.7 % pour celle de F1-RLT. Sauf que F3-MC nécessite beaucoup plus de temps de résolution que F1-RLT.

Nous passons aux résultats des relaxations linéaires des formulations pour les classes C, D et E. Comme il s'agit d'instances de moyennes et grandes tailles, aucune formulation de type F2 ainsi que la formulation F3-MC ne peut les résoudre même en terme de relaxation linéaire. Les tableaux des résultats sont les suivants:

Formulation	**Gap moyen**	**Gap maximal**	**Ecart Type**	**Temps Moyen**
F1-MTZ	13.17	27.46	9.27	0.18
F1-DL	3.96	10.91	3.24	0.28
F1-RLT	*0.16*	*0.48*	*0.21*	*4.34*
F3-SC	22.97	26.87	2.40	4.70
F3-SC-MTZ	22.97	26.87	2.40	5.03

Tableau 6.4: Tableau récapitulatif des résultats des relaxations linéaires des formulations compactes pour la classe C

Formulation	**Gap moyen**	**Gap maximal**	**Ecart Type**	**Temps Moyen**
F1-MTZ	19.88	22.11	1.66	0.16
F1-DL	6.05	7.42	1.05	0.24
F1-RLT	*0.14*	*0.35*	*0.12*	*7.94*

Tableau 6.5: Tableau récapitulatif des résultats des relaxations linéaires des formulations compactes pour la classe D

				F1-MTZ		F1-DL		F1-RLT		F3-SC		F3-SC-MTZ	
Inst.	n	M	Z^*	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps
C.01	500	625	1394	23.23	0.06	7.78	0.08	0.08	2.24	22.90	4.69	22.90	4.86
C.02	500	625	1430	23.42	0.06	7.02	0.08	0.21	1.92	22.81	4.56	22.81	4.80
C.03	500	625	1407	27.46	0.06	10.91	0.08	0.84	2.55	26.87	4.52	26.87	4.78
C.04	500	625	1433	22.36	0.06	6.83	0.06	0.14	2.20	21.86	4.58	21.86	5.11
C.05	500	625	1382	20.88	0.05	7.39	0.08	0.48	1.95	20.42	5.14	20.42	5.58
C.06	500	1000	1152	19.39	0.06	5.54	0.09	0.35	1.00	-	-	-	-
C.07	500	1000	1222	17.86	0.06	6.22	0.11	0.08	2.83	-	-	-	-
C.08	500	1000	1161	17.30	0.06	4.81	0.09	0.17	0.81	-	-	-	-
C.09	500	1000	1160	20.12	0.06	5.88	0.09	**0.00**	1.59	-	-	-	-
C.10	500	1000	1164	19.75	0.06	5.34	0.09	0.09	2.03	-	-	-	-
C.11	500	2500	752	10.01	0.11	2.01	0.25	**0.00**	2.05	-	-	-	-
C.12	500	2500	789	11.33	0.11	2.23	0.22	0.13	1.42	-	-	-	-
C.13	500	2500	789	9.97	0.11	2.05	0.19	0.26	1.53	-	-	-	-
C.14	500	2500	773	8.98	0.11	2.48	0.19	0.26	2.11	-	-	-	-
C.15	500	2500	770	11.18	0.11	2.67	0.22	0.13	1.59	-	-	-	-
C.16	500	12500	503	**0.00**	0.50	**0.00**	0.72	**0.00**	8.08	-	-	-	-
C.17	500	12500	499	**0.00**	0.50	**0.00**	0.72	**0.00**	10.58	-	-	-	-
C.18	500	12500	502	**0.00**	0.50	**0.00**	0.78	**0.00**	18.02	-	-	-	-
C.19	500	12500	504	**0.00**	0.50	**0.00**	0.70	**0.00**	11.24	-	-	-	-
C.20	500	12500	504	0.20	0.50	**0.00**	0.78	**0.00**	11.11	-	-	-	-
Moyenne				*13.17*	*0.18*	*3.96*	*0.28*	*0.16*	*4.34*	*22.97*	*4.70*	*22.97*	*5.03*
Ecart type				*9.27*		*3.24*		*0.21*		*2.40*		*2.40*	

Tableau 6.6: Performance des relaxations linéaires des formulations compactes sur la classe C

				F1-MTZ		F1-DL		F1-RLT	
Inst.	n	m	Z^*	Gap	Temps	Gap	Temps	Gap	Temps
D.01	1000	1250	2678*	20.52	0.16	7.16	0.22	0.28	3.09
D.02	1000	1250	2712*	22.11	0.14	7.42	0.22	0.28	4.53
D.03	1000	1250	2714	21.46	0.16	6.47	0.24	0.07	4.31
D.04	1000	1250	2701	20.94	0.14	7.16	0.22	0.18	3.89
D.05	1000	1250	2727	21.80	0.19	6.66	0.22	0.02	4.45
D.06	1000	2000	2313*	18.10	0.14	4.82	0.28	0.35	14.88
D.07	1000	2000	2329	18.40	0.17	5.15	0.24	0.09	9.95
D.08	1000	2000	2313	18.06	0.17	5.43	0.28	0.04	10.19
D.09	1000	2000	2327	18.12	0.17	4.58	0.23	0.04	11.92
D.10	1000	2000	2251	19.35	0.17	5.71	0.25	0.04	12.19
Moyenne				*19.88*	*0.16*	*6.05*	*0.24*	*0.14*	*7.94*
Ecart type				*1.66*		*1.05*		*0.12*	

Tableau 6.7 : Performance des relaxations linéaires des formulations compactes sur la classe D

				F1-MTZ		F1-DL		F1-RLT	
Inst.	n	m	Z^*	Gap	Temps	Gap	Temps	Gap	Temps
E.01	2500	3125	6842	22.60	0.88	7.24	1.19	0.01	30.00
E.02	2500	3125	7063	22.48	0.72	7.40	1.05	0.03	22.94
E.03	2500	3125	6959	23.06	0.67	7.12	1.06	**0.00**	14.94
E.04	2500	3125	6816*	21.96	0.74	6.75	1.17	0.15	33.84
E.05	2500	3125	7024*	22.87	0.64	7.18	1.03	0.15	28.91
Moyenne				*22.59*	*0.73*	*7.14*	*1.10*	*0.07*	*26.13*
Ecart type				*0.42*		*0.24*		*0.08*	

Tableau 6.8 : Performance des relaxations linéaires des formulations compactes sur la classe E

* : il s'agit de la meilleure borne supérieure calculée

Formulation	**Gap moyen**	**Gap maximal**	**Ecart Type**	**Temps Moyen**
F1-MTZ	22.59	23.06	0.42	0.73
F1-DL	7.14	7.40	0.24	1.10
F1-RLT	*0.07*	*0.15*	*0.08*	*26.13*

Tableau 6.9: Tableau récapitulatif des résultats des relaxations linéaires des formulations compactes pour la classe E

A partir des tableaux 6.6-6.8, On constate que:

- Les formulations de type F2, ainsi que F3-MC et F3-SC-RLT ne peuvent résoudre, en terme de relaxation linéaire, aucune instance de ces trois classes. Les formulations F3-SC et F3-SC-MTZ sont aussi limitées par la taille des instances puisqu'elles ne peuvent résoudre que les 5 premières instances de la classe C.

- Même si elle admet des temps de calcul extrêmement réduits, la formulation F1-MTZ fournit des gaps assez importants par rapport à F1-DL et F1-RLT.

- La formulation F1-RLT donne d'excellents résultats dans des temps de calcul assez raisonnables. En général, la performance d'une formulation augmente avec sa complexité, sauf que F1-RLT constitue une remarquable exception.

Après l'examen des résultats d'expériences sur les relaxations linéaires des formulations compactes du PCSTP, on conclut que:

La formulation F1-RLT admet la meilleure performance en terme de relaxation linéaire avec des temps de résolution assez réduits. Même si la formulation F3-MC donne de meilleurs bornes inférieurs, cette dernière ne peut résoudre que les instances de taille réduite.

6.6.2 Résolution exacte

Comme il s'agit de formulations valides du PCSTP, nous nous intéressons à la performance des formulations compactes que nous avons développées en terme de résolution exacte. Nous commencons par résoudre à l'optimum les 18 instances de la classe B.

Les détails des temps de résolution sont présentés dans le tableau 6.11 ci-dessous. Dans ce dernier, la colonne "B&C" correspond aux temps de résolution de la procédure exacte de type branch-and-cut que nous avons développé dans le chapitre 5.

Les temps moyens de résolution exacte des formulations compactes sont récapitulés dans le tableau suivant:

Formulation	**Temps Moyen**
F1-MTZ	4.46
F1-DL	4.60
F1-RLT	7.22
F3-MC	13.63
F3-SC	42.39
F3-SC-MTZ	*0.58*
F3-SC-RLT	527.10
B&C	0.52

Tableau 6.10: Tableau récapitulatif des temps moyens de résolution exacte des formulations compactes de la classe B

Inst.	F1-MTZ	F1-DL	F1-RLT	F3-SC	F3-SC-MTZ	F3-SC-RLT	F3-MC	B&C
B.01	0.25	0.06	0.09	0.75	0.14	4.47	0.61	0.21
B.02	19.67	10.41	17.59	2.42	0.34	8.88	0.44	0.24
B.03	0.53	0.52	3.48	0.25	0.09	3.38	0.17	0.21
B.04	2.75	0.98	10.92	3.72	0.56	87.95	1.91	0.42
B.05	0.06	0.03	0.09	2.66	0.22	86.52	0.80	0.26
B.06	0.78	1.03	3.22	12.45	0.47	336.34	8.19	0.43
B.07	0.36	0.17	0.36	6.74	0.50	224.20	3.56	0.82
B.08	2.36	0.88	2.63	3.45	0.38	15.78	2.30	0.32
B.09	6.06	10.44	10.84	0.77	0.25	13.67	1.86	0.58
B.10	0.14	0.17	0.16	24.56	0.50	757.39	9.99	0.54
B.11	11.14	6.50	22.89	29.83	0.48	1 868.83	9.77	0.25
B.12	0.53	0.08	0.14	18.20	0.48	2 349.99	7.70	0.32
B.13	2.77	1.59	3.72	9.70	1.00	503.41	14.77	0.95
B.14	6.72	8.33	9.33	5.98	0.41	327.61	5.31	0.73
B.15	0.80	1.83	4.19	15.53	1.06	391.86	12.95	0.82
B.16	1.33	0.97	1.83	232.39	1.77	> 6 000 *	44.52	0.51
B.17	0.11	0.06	0.22	12.02	0.64	1 453.38	16.06	0.45
B.18	24.05	38.88	38.34	381.77	1.16	> 6 000 *	104.61	1.28
Temps moyen	*4.47*	*4.61*	*7.22*	*42.40*	*0.58*	*527.10*	*13.64*	*0.52*

Tableau 6.11: Les temps de résolution exacte des instances de la classe B

Inst.	F1-MTZ	F1-DL	F1-RLT	F3-SC-MTZ	B&C
C.01	2.25	1.02	2.06	12.48	166.67
C.02	33.02	44.45	49.55	67.17	51.57
C.03	3 742.83	133.11	2 657.13	118.70	520.57
C.04	27.47	57.28	22.80	18.66	71.80
C.05	21.13	6.61	12.98	18.38	40.86
C.06	18.08	1.81	10.56	55.94	12.04
C.07	113.08	51.28	1 134.70	55.64	16.95
C.08	> 6 000 *	1 466.41	1 093.86	68.20	20.86
C.09	2.05	0.94	1.91	41.24	15.48
C.10	25.83	16.94	15.92	59.83	17.31
C.11	20.45	1.78	32.72	-	20.57
C.12	> 6 000 *	3.92	172.02	-	17.21
C.13	142.64	243.64	1316.74	-	28.79
C.14	14.02	99.41	39.47	-	12.17
C.15	73.80	108.44	20.59	-	20.70
C.16	18.17	119.23	27.94	-	-
C.17	70.89	16.81	1 450.67	-	-
C.18	73.16	20.63	16.45	-	-
C.19	32.16	6.61	1 527.91	-	-
C.20	79.31	21.72	126.19	-	-
Temps moyen	*250.57*	*121.10*	*486.61*	*51. 62*	*68.91*

Tableau 6.12: Les temps de résolution exacte des instances de la classe C

* : Temps de résolution > 6000 secondes
- : Pas de solution

En examinant, les tableaux ci-dessus 6.10 et 6.11 correpondant à la classe B, on constate que:

- Toutes les formulation de type F2 ne peuvent résoudre de manière exacte aucune instance de la classe B.
- Les deux formulations F3-SC et F3-SC-RLT ont les temps de résolution les plus importants. En plus, F3-SC-RLT a dépassé 6000 secondes de temps de calcul pour les instances B.16 et B.18.
- La formulation F3-SC-MTZ est la plus rapide. Ce qui est assez surprenant car cette formulation ne s'est pas distinguée en terme de relaxation linéaire. En effet, sa performance était assez similaire à celle de F3-SC.
- Les formulation de la famille F1 admettent de bons résultats et se trouvent en deuxième position après la formulation F3-SC-MTZ.

Suite à ces tests sur la classe de petites instances B, nous passons à la résolution exacte des 20 instances de la classe C. A cause de leur taille atteignant les 500 noeuds et les 12500 arêtes, aucune formulation de la famille F2 ne peut résoudre de manière exacte les instances de la classe C. Nous appliquons toutes les formulations de type F1 et F3 et nous fixons le temps maximal de résolution à 6000 secondes.

Les temps de résolution exacte des formulations sont détaillés dans le tableau 6.12. Un récapitulatif des temps moyens est présenté dans le tableau 6.13 suivant:

Formulation	**Temps Moyen**
F1-MTZ	250,57
F1-DL	*121,10*
F1-RLT	486,60
F3-SC-MTZ	51,623

Tableau 6.13: Tableau récapitulatif des temps moyens de résolution exacte des formulations compactes de la classe C

Partant des résultats des tableaux 6.12 et 6.13, nos constatations sont les suiavntes:

- Les formulations F3-SC, F3-SC-RLT et F3-MC ne peuvent résoudre à l'optimum aucune instance de la classe C.
- Par rapport aux autres formulations de type F3, F3-SC-MTZ admet une performance exceptionnelle en terme de temps de calcul sauf qu'elle reste limitée par la taille des instances: elle ne peut résoudre que les 10 premières instances de la classe C.
- La formulation F1-MTZ résoud à l'optimum seulement 18 instances avec un temps moyen de l'ordre 250 secondes.
- La formulation F1-DL est la plus performante suivie de F1-RLT. En plus, elle résoud à l'optimum toutes les instances contrairement à F3-SC-MTZ qui ne peut résoudre que les 10 premières instances.

Après avoir testé la performance des formulations compactes en terme de résolution exacte sur une trentaine d'instances de taille différentes, on peut conclure que:

La formulation F3-SC-MTZ est la plus rapide sur les petites et moyennes instances (jusqu'à 500 noeuds et 1000 arcs). Pour les grandes instances, F1-DL suivi de F1-RLT sont les formulations les plus performantes.

Formulations compactes vs procédure branch-and-cut

Nous rappelons que nous avons déjà développé une procédure exacte lors du chapitre 5 et que nous avons déjà résolu de manière exacte les classes B et C. Partant des tableaux des résulats, nous pouvons conclure que pour la classe B qui ne contient que des petites instances, la performance de la procédure branch-and-cut est assez similaire à celle de F3-SC-MTZ. Pour la classe C, la procédure exacte admet une assez bonne performance sauf qu'elle est assez vite limitée par la taille des instances. En effet, elle ne peut résoudre que seulement les 15 premières instances de la classe C.

6.6.3 Résolution approchée

Le solveur Cplex offre la possibilité de résoudre de manière approchée un programme en nombre entier. En effet, il suffit de lui fixer une tolérance entre 0 et 1 qui correspond à un gap lors du branchement sa procédure interne. Une

tolérance égale à 0 (par défaut 10^{-4}) correspond à une résolution exacte du problème en nombre entier. Il est évident que plus la valeur de la tolérance fixée est réduite, plus le temps de résolution augmente. Plusieurs valeurs de tolérance ont été testées avec les différentes formulations pour les classes d'instances et les meilleures performances ont été obtenues avec une tolérance de 0.5% pour la classe C et de 0.3% pour les classes D et E. Les détails des résultats sont disponibles dans les tableaux 6.16-6.18. La signification des colonnes est la suivante:

- $Gap = 100 \times \frac{UB - Z^*}{Z^*}$, où UB est une borne supérieure et Z^* est la solution optimale,

- $Temps$: le temps CPU nécéssaire au calcul de UB en secondes.

- HSPI : correspond aux résultats de l'heuristique HSPI que nous avons déjà proposée dans le chapitre 4.

Les tableaux récapitulatifs de la performance des formulations compactes en terme de résolution approchée sont les suivants:

Formulation	**Gap moyen**	**Gap maximal**	**Ecart Type**	**Temps Moyen**
F1-MTZ	0.11	0.29	0.12	35.97
F1-DL	*0.08*	*0.39*	*0.11*	*35.31*
F1-RLT	*0.06*	*0.36*	*0.10*	*278.85*
F3-SC-MTZ	0.10	0.29	0.12	43.42

Tableau 6.14: Tableau récapitulatif de la performance des formulations compactes en terme de résolution approchée pour la classe C

Formulation	**Gap moyen**	**Gap maximal**	**Ecart Type**	**Temps Moyen**
F1-MTZ	0.11	0.22	0.08	324.73
F1-DL	*0.09*	*0.18*	*0.07*	*731.41*
F1-RLT	0.12	0.22	0.08	1 107.73

Tableau 6.15: Tableau récapitulatif de la performance des formulations compactes en terme de résolution approchée pour la classe D

	F1-MTZ		F1-DL		F1-RLT		F3-SC-MTZ		HSPI	
Inst.	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps
C.01	0.07	1.67	0.15	0.84	**0.00**	2.08	0.29	9.80	0.15	9.85
C.02	**0.00**	32.80	0.07	44.61	0.07	48.64	0.28	57.80	0.21	17.01
C.03	0.29	18.45	0.14	30.83	0.36	581.36	0.22	85.72	0.36	31.56
C.04	0.07	18.56	0.14	17.69	0.07	4.14	0.07	12.75	**0.00**	46.81
C.05	0.22	14.09	0.15	6.34	**0.00**	3.73	**0.00**	11.08	0.37	14.87
C.06	0.18	11.36	**0.00**	1.72	**0.00**	3.14	0.09	44.16	**0.00**	1.54
C.07	**0.00**	112.72	**0.00**	52.05	0.08	847.83	0.08	50.00	0.25	2.42
C.08	**0.00**	47.50	0.26	81.00	0.17	10.89	**0.00**	68.16	**0.00**	1.98
C.09	0.26	2.03	**0.00**	0.94	**0.00**	2.06	**0.00**	41.30	**0.00**	1.87
C.10	0.17	18.81	**0.00**	16.83	**0.00**	16.08	**0.00**	53.48	**0.00**	2.42
C.11	0.13	17.45	**0.00**	1.67	**0.00**	32.55	-	-	**0.00**	1.46
C.12	0.13	33.11	0.13	2.69	**0.00**	174.72	-	-	**0.00**	0.84
C.13	**0.00**	63.30	0.26	108.81	0.26	674.39	-	-	0.13	1.26
C.14	**0.00**	5.95	**0.00**	58.78	0.13	14.88	-	-	**0.00**	1.03
C.15	0.26	67.61	0.39	97.55	**0.00**	20.69	-	-	**0.00**	1.24
C.16	**0.00**	18.31	**0.00**	118.14	**0.00**	27.94	-	-	-	-
C.17	**0.00**	70.84	**0.00**	16.89	**0.00**	1452.59	-	-	-	-
C.18	**0.00**	72.92	**0.00**	20.59	**0.00**	16.38	-	-	-	-
C.19	0.40	12.38	**0.00**	6.58	**0.00**	1517.08	-	-	-	-
C.20	**0.00**	79.45	**0.00**	21.70	**0.00**	125.77	-	-	-	-
Moyenne	*0.11*	*35.97*	*0.08*	*35.31*	*0.06*	*278.85*	*0.10*	*43.42*	*0.10*	*9.08*
Ecart type	*0.12*		*0.11*		*0.10*		*0.12*		*0.13*	

Tableau 6.16 : Performance des formulations compactes en terme de résolution approchée de la classe C

	F1-MTZ		F1-DL		F1-RLT		HSPI	
Inst.	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps
D.01	0.04	156.63	0.07	285.05	0.04	214.74	0.11	211.27
D.02	0.07	1946.48	0.04	355.34	0.11	3466.56	0.18	149.06
D.03	0.11	26.08	0.18	71.44	0.22	24.23	0.00	50.63
D.04	0.22	225.63	0.15	246.17	0.15	422.17	0.26	102.16
D.05	0.22	3.78	0.00	1.67	0.00	3.05	0.26	91.81
D.06	-	-	0.17	3815.22	-	-	0.17	10.69
D.07	0.21	318.00	0.00	2151.25	0.13	5212.05	0.09	7.50
D.08	0.13	6.11	0.13	3.55	0.13	300.69	0.04	9.05
D.09	0.00	190.17	0.13	196.33	0.21	278.39	0.00	13.38
D.10	0.09	49.69	0.00	188.09	0.22	47.70	0.13	9.20
Moyenne	*0.11*	*324.73*	*0.09*	*731.41*	*0.12*	*1107.73*	*0.12*	*65.47*
Ecart type	*0.08*		*0.07*		*0.08*		*0.10*	

Tableau 6.17: Performance des formulations compactes en terme de résolution approchée de la classe D

	F1-MTZ		F1-DL		F1-RLT		HSPI	
Inst.	Gap	Temps	Gap	Temps	Gap	Temps	Gap	Temps
E.01	0.22	185.63	0.09	9.75	0.15	119.59	-	-
E.02	0.08	1038.88	0.08	381.52	0.06	2199.86	-	-
E.03	0.22	153.05	0.04	7.91	0.14	60.00	-	-
E.04	0.07	5682.91	0.10	196.06	-	-	-	-
E.05	-	-	0.07	219.75	-	-	-	-
Moyenne	*0.15*	*1765.11*	*0.08*	*163.00*	*0.12*	*793.15*	-	-
Ecart type	*0.08*		*0.02*		*0.05*		-	-

Tableau 6.18 : Performance des formulations compactes en terme de résolution approchée de la classe E

- : pas de solution donnée par CPLEX au bout de 6000 secondes de calcul

Formulation	Gap moyen	Gap maximal	Ecart Type	Temps Moyen
F1-MTZ	0.15	0.22	0.08	1 765.11
F1-DL	*0.08*	*0.10*	*0.02*	*163.00*
F1-RLT	0.12	0.15	0.05	793.15

Tableau 6.19: Tableau récapitulatif de la performance des formulations compactes en terme de résolution approchée pour la classe E

En évaluation de la performance des formulations développées en terme de résolution approchée, nous constatons que:

- Même si la formulation F3-SC-MTZ est la plus performante pour la résolution exacte des instances de taille réduite, sa performance est assez limitée en terme de résoltuin approchée.
- Les formulations F1-MTZ et F1-RLT ne peuvent pas résoudre de manière approchée plusieurs instances de grande taille telles que D.06, E.04 et E.05. Malgrès leurs bons résultats, elles sont moins robustes que la formulation F1-DL.

Notre conclusion est que *F1-DL est la formulation la plus performante en terme de résolution approchée du PCSTP.*

Formulations compactes vs l'heuristique HSPI

L'inconvenient majeur de l'heuristique HSPI est qu'elle est limitée en terme de taille des instances. En particulier, aucune instance de la classe E ne peut être résolue par HSPI. Pour les classes C et D, HSPI s'est distinguée par des gaps de l'ordre de 0.1% en des temps de résolution extrêmement réduits. Mais, globalement, la formulation F1-DL est plus perfomante que l'heuristique HSPI.

6.6.4 Preprocessing et résolution exacte des problèmes de grandes tailles

Dans la littérature, les classes D et E du *SteinLib* sont connues comme des classes d'instances de grande taille qui sont assez difficiles à résoudre. Lors de ce travail, nous proposons de les résoudre de manière exacte en appliquant les formulations compactes que nous avons développées. C'est la

taille de leurs instances qui justifie le recours à une procédure de réduction du graphe (preprocessing) avant la résolution exacte des instances réduites.

Il s'agit du preprocessing basé sur la relaxation linéaire tel qu'il est décrit dans le paragraphe 5.3.2. Pour avoir un preprocessing efficace, nous avons choisit de faire appel aux formulations les plus performantes en terme de relaxation linéaire et de résolution approchée , i.e celles qui donnent les meilleures bornes inférieures (LB) et supérieures (UB). Ainsi, suite aux conclusions des paragraphes 6.6.1 et 6.6.3, nous avons appliqué la relaxation linéaire de la formulation F1-RLT. Et nous avons utilisée les bornes supérieures de la formulation F1-DL des tableaux 6.17 et 6.18. Pour évaluer la performance du preprocessing dont les résultats sont présentés dans les tableaux 6.21 et 6.22, on calcule les indicateurs suivants:

- PDA : le pourcentage des arcs enlevés.
- PDN : le pourcentage des noeuds enlevés.
- PFN : le pourcentage des noeuds dont les variables sont fixées à 1.

Grâce au test d'optimalité (si $UB = \lceil LB \rceil$, alors c'est l'optimum), on trouve l'optimum pour l'instances D.05. Pour les 2 classes des instances réduites par le preprocessing, le récapitulatif des résultats est le suivant:

PDA moyen	12.98%
PDN moyen	5.89%
PFN moyen	11.80%

Tableau 6.20: Tableau récapitulatif de la performance du preprocessing des classes D et E

Comme il s'agit de graphes très peu denses, ces résultats sont extrêment intéréssants et prouve la bonne performance du preprocessing proposé. En tenant compte de la bonne performance des formulations F1-RLT et F1-DL en terme de résolution approché et de relaxation linéaire, plusieurs stratégies ont été testées. Celle que nous présentons est la plus efficace en terme du compromis entre la qualité des résultats et les temps de calcul.

Inst.	n	m	PDA	PDN	PFN	Temps F1-DL	Temps Total
D.01	1000	1250	0.64	0.00	1.30	-	-
D.02	1000	1250	0.40	0.00	2.60	> 6000	> 6000
D.03	1000	1250	4.36	3.50	8.40	82.37	158.46
D.04	1000	1250	0.64	0.00	2.20	118.31	369.73
D.06	1000	2000	0.90	0.90	0.00	-	-
D.07	1000	2000	42.75	16.70	34.70	> 6000	> 6000
D.08	1000	2000	19.80	6.00	24.40	102.50	122.90
D.09	1000	2000	20.43	6.30	22.10	51.93	267.27
D.10	1000	2000	57.78	26.50	40.40	13.96	209.16
Moyenne			*16.41*	*6.66*	*15.12*	*73.81*	*225.51*

Tableau 6.21: Résultats du preprocessing et de la résolution exacte de la classe D

Inst.	n	m	PDA	PDN	PFN	Temps F1-DL	Temps Total
E.01	2500	3125	2.69	1.88	3.96	13.37	46.02
E.02	2500	3125	1.09	0.72	3.16	354.70	775.70
E.03	2500	3125	29.62	19.96	21.04	3.96	36.25
E.04	2500	3125	0.38	0.00	0.92	-	-
E.05	2500	3125	0.27	0.00	0.00	-	-
Moyenne			*6.81*	*5.64*	*7.27*	*124.01*	*285.99*

Tableau 6.22: Résultats du preprocessing et de la résolution exacte de la classe E

- : CPLEX est bloqué et ne donne aucun résultat
> 6000 : CPLEX n'a rien donné au bout de 6000 secondes de calcul

Suite au preprocessing, nous appliquons la formulation F1-DL pour la résolution exacte des problèmes réduits puisqu'elle a prouvé sa bonne performance lors de la résolution exacte des classes B et C dans le paragraphe 7.6.2. Nous rapportons les temps de calcul dans les tableaux 6.21 et 6.22 dont la signification des colonnes est la suivante:

- Temps F1-DL = temps de la résolution exacte de l'instance réduite par F1-DL.

- Temps Total = temps de calcul de UB + temps de calcul de LR + temps des tests du preprocessing + Temps F1-DL.

Nous avons constaté que le solveur CPLEX se bloque et ne peut pas résoudre de manière exacte les 4 instances réduites D01, D06, E04 et E05. Pour ce qui concerne les instances D02 et D07, au bout de 6000 secondes, CPLEX n'obtient pas de solution exacte. Même si la méthode exacte de type branch and cut, que nous avons présenté dans le chapitre 5, est plus rapide pour la classe D, elle ne peut résoudre aucune instance de la classe E.

Ainsi, nous avons résolu de manière exacte 9 des 15 instances des classes D et E dans un temps moyen de 220 secondes. Pour les 6 instances restantes, nous avons obtenu des solutions approchées avec un gap moyen de 0,07 % par rapport à l'optimum. En fixant la tolérance à 0.1%, toutes les instances sont résolues.

6.7 Conclusion

Dans ce chapitre, nous nous sommes intéressés au développement de formulations compactes dont le nombre de variables et de contrainte est polynomial. Une première famille F1 de trois formulations inspirées des contraintes de type MTZ (Miller-Tucker-Zemlin [158]) a été présentée. Une deuxième famille F2 de formulations basées sur les chemins a été aussi exposée. Nous avons aussi développé une troisième famille de formulations F3 basée sur la notion du flot. Une étude théorique des relations de dominance entre les relaxations linéaires de ces différentes formulations a été introduite.

En utilisant le solveur de programmes linéaires CPLEX, nous avons effectué une étude expérimentale exhaustive afin de comparer la performance des formulations développées en terme de relaxation linéaire, résolution exacte et approchée. Les tests étaient effectués sur les 4 classes (B, C, D et E)

de 53 instances benchmark du *SteinLib* [131]. Nous avons constaté que les résultats des formulations de la famille F2 sont assez limités par la taille des instances alors que les formulations F1-DL et F1-RLT se sont révélées très performantes. En effet, nous avons résolu 47 des 53 instances à l'optimum. Pour les 6 instances restantes, nous avons obtenu des solutions approchées avec un gap moyen de 0,07 % par rapport à l'optimum. Notons que grâce à ces formulations compactes, des instances de 2500 noeuds sont résolues à l'optimum en des temps satisfaisants.

Chapitre 7

Conclusion Générale

Dans cette thèse, nous nous sommes intéréssés à la modélisation et la résolution du problème de Steiner avec collecte de prix (Prize Collecting Steiner Tree Problem PCSTP) et à une version généralisée du PCSTP que nous avons appelé le problème de Steiner avec collecte de prix généralisé (Generalized Prize Collecting Steiner Tree Problem GPCSTP). L'intérêt théorique de l'étude de ce problème provient essentiellement du fait qu'il constitue un cadre unificateur d'une riche classe de problèmes d'optimisation NP-difficiles définis sur les structures arborescentes. Il est aussi d'un intérêt pratique incontestable vu ses applications dans le secteur des télécommunications.

Afin de proposer des bornes inférieures pour le GPCSTP, nous avons commencé par développer une formulation mathématique basée sur la recherche d'une structure arborescente avec des conditions additionnelles. Grâce à cette formulation, nous avons pu appliquer la technique de la relaxation lagrangienne avec quatre formes différentes. Nous avons résolu les duaux lagrangiens en appliquant neuf variantes différentes de l'algorithme du sous-gradient, l'algorithme du volume, deux variantes de la méthode VTVM et un algorithme exact basé sur la génération de coupes avec stabilisation. Une analyse théorique comparant les différentes relaxations a été effectuée et une étude expérimentale comparant les différentes approches de résolution du dual lagrangien a été présentée pour 25 instances générées aléatoirement. Nous avons trouvé que la version ADS de l'algorithme du sous-gradient donne la meilleure performance. Bien que la solution primale obtenue par ADS n'est pas nécessairement réalisable, nous l'avons exploité pour appliquer un preprocessing et aussi pour construire une solution réalisable et par conséquent

une borne supérieure pour le PCSTP.

Afin d'obtenir des solutions approchées pour le PCSTP, nous avons développé deux versions d'une heuristique basée sur la décomposition. En effet, cette heuristique décompose le problème en deux sous-problèmes : un problème d'arbre de poids minimal et un PCSTP défini sur un arbre. Chacun de ces deux problèmes est résolu par un algorithme exact. Nous avons démontré que résoudre le PCSTP sur un graphe, qui est un arbre, est toujours NP-difficile sauf qu'on peut désormais le résoudre de façon pseudo-polynômiale. L'idée de l'heuristique est d'appliquer un pruning exact sur un arbre en résolvant un problème de plus court chemin avec une contrainte de capacité.

Toujours pour le PCSTP, nous avons développé une formulation mathématique basée sur les coupes auquel nous avons ajouté des contraintes valides (modèle EDCF). Comme il s'agit d'un modèle comportant un nombre exponentiel de contraintes, nous avons proposé un algorithme exacte de génération de contraintes. Avant de passer à la résolution exacte, nous avons développé des procédures de preprocessing qui sert à réduire significativement la taille des instances en utilisant l'excellente borne supérieure de l'heuristique. Grâce à cette procédure exacte de type branch-and-cut, des instances ayant jusqu'à 1000 noeuds ont été résolu à l'optimum dans des temps raisonnables.

La dernière partie de cette thèse a été consacrée au développement de trois familles de formulations compactes pour le PCSTP. Il s'agit de formulations se caractérisant par un nombre polynomial de contraintes et de variables. Nous avons proposé une comparaison des relaxations linéaires des formulations proposées. Une étude expérimentale exhaustive a été effectuée pour tester la performance de ces formulations en termes de qualités des relaxations linéaires et aussi en résolution exacte et approchée. Ainsi, nous avons résolu de manière exacte des instances avec 2500 noeuds en des temps assez satisfaisants.

Les travaux issus de cette thèse ont fait l'objet d'une première publication qui a apparu dans une revue internationale ([102]) et d'un deuxième papier en cours de révision ([103]). La dernière partie de cette thèse fait l'objet d'un papier en cours de préparation qui sera très bientôt soumis pour publication.

Concernant les recherches futures, une approche exacte basée sur la décomposition de Benders peut être proposée pour le PCSTP. Avec l'impressionnant développement exponentiel des solveurs de programmes linéaires, la

mise au point de formulations compactes pour les problèmes d'optimisation combinatoire est une excellente alternative des méthodes classiques de résolution. Nous comptons étendre les formulations proposées afin de modéliser et de résoudre de nouvelles variantes du GMST [84] et du PCSTP [26] et [36].

Bibliographie

[1] Ahuja R.K., J.B. Orlin, D. Sharma (2003), A composite very large-scale neighborhood structure for the capacitated minimum spanning tree problem, *Operations Research Letters* 31, 185-194.

[2] Ahuja R.K., T.L. Magnanti, and J.B. Orlin (1993), *Network flows: theory, algorithms, and applications*. Prentice Hall, Upper Saddle River, New Jersey.

[3] Amberg A., W. Domschke, S. Voβ (1996), Capacitated minimum spanning trees: algorithms using intelligent search, *Combinatorial Optimization: Theory and Practice* 1, 9-40.

[4] Angelopoulos S. (2006), The node-weighted Steiner problem in graphs of restricted node weights, *Lecture Notes in Computer Science* 4059, 208-219.

[5] Anstreicher K.M., L. Wolsey (1993), On dual solutions in subgradient optimization, Technical report, Department of Management Sciences, University of Iowa. Iowa City.

[6] Arya S., H. Ramesh (1998), 2.5-factor approximation algorithm for the k-MST problem, *Information Processing Letters* 65, 117-118.

[7] Balas E. (1989), The prize-collecting traveling salesman problem, *Networks* 19, 621-636.

[8] Bahiense L., F. Barahona, O. Porto (2003), Solving Steiner tree problems in graphs with Lagrangian relaxation, *Journal of Combinatorial Optimization* 7, 259-282.

[9] Bahiense L., N. Maculan, C. Sagastizábal (2002), The volume algorithm revisited: relation with bundle methods, *Mathematical Programming* 94, 41-69.

[10] Barahona F., R. Anbil (2000), The volume algorithm : producing primal solutions with a subgradient method, *Mathematical Programming* 87, 385-399.

[11] Barahona F., L. Ladányi (2006), Branch and cut based on the volume algorithm: Steiner trees in graphs and max-cut , *RAIRO Recherche Opérationnelle* 40, 53-73.

[12] Barreiros J., E. Costa (2003), An hierarchic genetic algorithm for computing (near) optimal euclidean Steiner trees, *Workshop on Application of Hybrid Evolutionary Algorithms to NP-Complete Problems GECCO2003*, Chicago.

[13] Bazaraa M.S., J.J. Goode (1977), The traveling salesman problem: a duality approach, *Mathematical Programming* 13, 221-237.

[14] Bazlamaçci C.F., K.S. Hindi (2001), Minimum-weight spanning tree algorithms: a survey and empirical study, *Computers & Operations Research* 28, 767-785.

[15] Beasley J.E. (1992), A heuristic for euclidean and rectilinear Steiner problems, *European Journal of Operational Research* 58, 284-292.

[16] Beasley J.E. (1989), An SST-based algorithm for the Steiner problem in graphs, *Networks* 19, 1-16.

[17] Bienstock D., M. Goemans, D. Simchi-Levi, D. Williamson (1993), A note on the prize collecting traveling salesman problem, *Mathematical Programming* 59, 413-420.

[18] Blum C. (2002), Ant colony optimization for the edge-eighted k-cardinality tree problem, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002* New York, 35-42.

[19] Blum C., M.J. Blesa (2005), New metaheuristic approaches for the edge-weighted k-cardinality tree problem, *Computers and Operations Research* 32, 1355-1377.

[20] Blum C., M. Ehrgott (2003), Local search algorithms for the k-cardinality tree problem, *Discrete Applied Mathematics* 128, 511-540.

[21] Borndörfer R., C. Fereira, A. Martin (1998), Decomposing matrices into blocks, *SIAM Journal on Optimization* 9, 236-269.

[22] Camerini P.M., L. Frata, F. Maffioli (1975), On improving relaxation methods by modified gradient techniques, *Mathematical Programming Study* 3, 26-34.

[23] Canuto S.A., M.G.C. Resende, C.C. Ribeiro (2001), Local search with perturbations for the prize-collecting Steiner tree problem in graphs, *Networks* 38, 50-58.

[24] Cayley A. (1897), *Collected Mathematical Papers*, Cambridge University Press, Vol. I-XIII.

[25] Chandy K.M., T. Lo (1973), The capacitated minimum spanning tree, *Networks* 3, 173-181.

[26] Chapovska O., A. P. Punnen (2006), Variations of the Prize-Collecting Steiner Tree Problem, *Networks* 47, 199-205.

[27] Charikar M., C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, M. Li, (1999) , Approximation algorithms for directed Steiner problems, *Journal of Algorithms* 33, 73-91.

[28] Chauve C. (2000), *Structures arborescentes : problèmes algorithmiques et combinatoires*, Thèse présentée à l'Université Bordeaux I.

[29] Chekuri C., G. Evenb, G. Kortsarzc (2006), A greedy approximation algorithm for the group Steiner problem, *Discrete Applied Mathematics* 154, 15-34.

[30] Chopra S., E.R. Gorres, M.R. Rao (1992), Solving the Steiner tree problem in graphs using branch-and-cut, *ORSA Journal on Computing* 4, 320-335.

[31] Chopra S., C.Y. Tsai (2001), Polyhedral approaches for the Steiner tree problem on graphs, *Combinatorial Optimization* 11, 175-201.

[32] Chudak F.A., T. Roughgarden, D.P. Williamson (2004), Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation, *Mathematical Programming* 100, 411-421.

[33] Cieslik D., A. Dress, K.T. Huber, V. Moulton (2002), Embedding complexity and discrete optimization II: a dynamical programming approach to the Steiner tree problem, *Annals of Combinatorics* 6, 275-283.

[34] Cockayne E.J., Z.A. Melzak (1968), Steiner's problem for set terminals, *Quarterly Applied Mathematics* 26, 213-218.

[35] Cole R., R. Hariharan, M. Lewenstein, E. Porat (2001), A faster implementation of the Goemans-Williamson clustering algorithm, *SODA01: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, 17-25.

[36] Cordone R., M. Trubian (2008), A relax-and-cut algorithm for the knapsack node weighted Steiner tree problem, *Asia-Pacific Journal of Operational Research* 25, 373-391.

[37] Cordone R., M. Trubian (2006), An exact algorithm for the node weighted Steiner tree problem , *4OR: A Quarterly Journal of Operations Research* 4, 124-144.

[38] Cornuejols G., M.L. Fisher, G.L. Nemhauser (1977), Location of bank accounts to optimize float: analytic study of exact and approximate algorithms, *Management Science* 23, 789-810.

[39] Costa A.M., J.F. Cordeau, G. Laporte (forthcoming), Models and branch-and-cut algorithms for the Steiner tree problem with revenues, budget and hop constraints, Networks.

[40] Costa A.M., J.F. Cordeau, G. Laporte (2006), Steiner tree problems with profits, *INFOR* 14, 99-115.

[41] Courant R., H. Robbins (1941), *What is mathematics?*, London: Oxford University Press.

[42] Crowder H. (1976), *Computational improvements for subgradient optimization*, In: Symposia Mathematica XIX, Academic Press, London, 357-372.

[43] Dantzig G.B., D.R. Fulkerson, S.M. Johnson (1954), Solution of a large-scale traveling salesman problem, *Operations Research* 2, 393-410.

[44] De Souza C.C., Ribeiro C.C. (1993), Heuristics for the minimum rectilinear Steiner tree problem: new algorithm and a computational study, *Discrete Applied Mathematics* 45, 205-220.

[45] Desrochers M., G. Laporte (1991), Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints, *Operations Research Letters* 10, 27-36.

[46] Dijkstra E.W. (1959), A note on two problems in connexion wih graphs, *Numerische Mathematik* 1, 269-271.

[47] Dreyer D.R., M.L. Overton (1998), Two heuristics for the Euclidean Steiner tree problem, *Journal of Global Optimization* 13, 95-106.

[48] Dreyfus S.E., R.A. Wagner (1971), The Steiner problem in graphs, *Networks* 1, 195-207.

[49] Dror M., M. Haouari (2000), Generalized Steiner problems and other variants, *Journal of Combinatorial Optimization* 4, 415-436,

[50] Dror M., M. Haouari, J. Chaouachi (2000), Generalized spanning trees, *European Journal of Operational Research* 120, 583-592.

[51] Du D.Z., B. Lu, H. Ngo, P.M. Pardalos (2001), Steiner tree problems, *Encyclopedia of Optimization* 5, 227-290.

[52] Duin C.W., A. Volgenant (1987), Some generalizations of the Steiner problem in graphs, *Networks* 17, 353-364.

[53] Duin C.W., A. Volgenant, S. Voβ (2004), Solving group Steiner problems as Steiner problems, *European Journal of Operational Research* 154, 323-329.

[54] Edmonds J. (1971), Matroids and the greedy algorithm, *Mathematical Programming* 1, 127-136.

[55] Engevall S., M. Göthe-Lundgren, P. Värbrand (1998), A strong lower bound for the node weighted Steiner tree problem, *Networks* 31, 11-17.

[56] Ehrgott M., J. Freitag (1996), k-tree/k-subgraph: a program package for minimal weighted k-cardinality trees and subgraphs, *European Journal of Operational Research* 93, 224-225.

[57] Ehrgott M., J. Freitag, H.W. Hamacher, F. Maffioli (1997), Heuristics for the k-cardinality tree and subgraph problem, *Asia Pacific Journal of Operational Research* 14, 87-114.

[58] Esau L.R., K.C. Williams (1966), On teleprocessing system design. Part II: a method for approximating the optimal network, *IBM Systems Journal* 5, 142-147.

[59] Esbensen H. (1995), Computing near-optimal solutions to the Steiner problem in graphs using a genetic algorithm, *Networks* 26, 173–1855.

[60] Even S., A. Itai, A. Shamir (1976), On the complexity of timetable and multicommodity fow problems, *SIAM Journal on Computing* 5, 88-124.

[61] Feigenbaum J., C.H. Papadimitriou, S. Shenker (2001), Sharing the cost of multicast transmissions, *Journal of Computer and System Sciences* 63, 21-41.

[62] Feillet D., P. Dejax, M. Gendreau (2005), Traveling salesman problems with profits, *INFORMS Transportation Science* 39, 188-205.

[63] Feremans C., M. Labbé, G. Laporte (2004), The generalized minimum spanning tree problem: polyhedral analysis and branch-and-cut algorithm, *Networks* 43, 71-86.

[64] Feremans C., M. Labbé, G. Laporte (2002), A comparative analysis of several formulations for the generalized minimum spanning tree problem, *Networks* 39, 29-34.

[65] Ferreira C.E., F.M.O. Filho (2006), Some formulations for the group steiner tree problem, *Discrete Applied Mathematics* 154, 1877-1884.

[66] Fischetti M. (1991), Facets of two Steiner arborescence polyhedra, *Mathematical Programming* 51, 401-419.

[67] Fischetti M., H.W. Hamacher, K. Jörnsten, F. Maffioli (1994), Weighted k-cardinality trees: complexity and polyhedral structure, *Networks* 24, 11-21.

[68] Fisher M.L. (1985), An applications oriented guide to Lagrangean relaxation, *Interfaces* 15, 10-21.

[69] Fisher M.L. (1981), The Lagrangian relaxation method for solving integer programming problems, *Management Science* 27, 1-18.

[70] Ford L.R., D.R Fulkerson (1974), *Flows in Networks*, Princeton University Press.

[71] Foulds L.R., H.W. Hamacher, J. Wilson (1998), Integer programming approaches to facilities models with forbidden areas, *Annals of Operations Research* 81, 405-417.

[72] Fredman M.L., R.E. Tarjan (1987), Fibonacci heaps and their uses in improved network optimisation algorithms, *Journal of the Association for Computing Machinery* 34, 596-615.

[73] Ganley J.L., J.P. Cohoon (1996), Rectilinear steiner trees on a checkboard, *ACM Transactions on Design Automation of Electronic Systems* 4, 512-522.

[74] Garey M.R., D.S. Johnson (1977), The rectilinear Steiner problem is NP-complete, *SIAM Journal of Applied Mathematics* 32, 826-834.

[75] Garg N. (2005), Saving an ε: a 2-approximation for the k-MST problem in graphs, *Proceedings of 37th ACM Symposium on Theory of Computing*, 396-402.

[76] Garg N. (1996), A 3-approximation for the minimum tree spanning k vertices, *Proceedings of 37th IEEE Symposium on Foundations of Computer Science,* 302- 309.

[77] Garg N., D. Hochbaum (1997), An $O(\log k)$ approximation algorithm for the k minimum spanning tree problem in the plane, *Algorithmica* 18, 111-121.

[78] Gavish B. (1983), Formulations and algorithms for the capacitated minimal directed tree problem, *Journal of the Association of Computing Machinery* 30, 118-132.

[79] Gavish B. (1982), Topological design of centralized computer networks formulations and algorithms, *Networks* 12, 355-377.

[80] Geoffrion A.M. (1974), Lagrangian relaxation and its uses in integer programming, *Mathematical Programming Study* 2, 82-114.

[81] Geoffrion A.M., R. McBride (1978), Lagrangian relaxation applied to capacitated facility location problems, *American Institute of Industrial Engineers Transactions* 10, 40-47.

[82] Goemans M.X., D.P. Williamson (1997), *The primal-dual method for approximation algorithms and its application to network design problems*, in D.S. Hochbaum (editor), Approximation Algorithms for NP-Hard Problems, 144-191, PWS Publishing Company, Boston.

[83] Goffin J.L. (1977), On convergence rates of subgradient optimization methods, *Mathematical Programming* 13, 329-347.

[84] Golden B., S. Raghavan, D. Stanojevic (2008), The prize-collecting generalized minimum spanning tree problem, *Journal of Heuristics* 14, 69-93.

[85] Gordeev E.N., O.G. Tarastsov (1993) , Steiner problem: a review, *Diskretn Mat* 5, 3-28.

[86] Gouveia L. (1995), A 2^{nd} constraint formulation for the capacitated minimum spanning tree problem, *Operations Research* 43, 130-141.

[87] Gouveia L. (1995), Using the Miller-Tucker-Zemlin constraints to formulate minimal spanning trees with hop constraints, *Computers & Operations Research* 22, 959-970.

[88] Gouveia L. (1993), A comparison of directed formulations for the capacitated minimum spanning tree problem, *Telecommunications Systems* 1, 51-76.

[89] Gouveia L., P. Martins (1995), An extended flow based formulation for the capacitated minimal spanning tree, *The Third ORSA Telecommunications Conference*, Boca Raton.

[90] Gouveia L., J. Pires (1999), The asymmetric traveling salesman problem and a reformulation of the Miller-Tucker-Zemlin constraints, *European Journal of Operational Research* 112, 134-146.

[91] Graham R.L., P. Hell (1985), On the history of the minimum spanning tree problem, *Annals of the History of Computing* 7,43-57.

[92] Guha S., S. Khuller (1999), Improved methods for approximating node weighted Steiner trees and connected dominating sets, *Information and computation* 150, 57-74.

[93] Guignard M., S. Kim (1987), Lagrangian decomposition: a model yelding stronger Lagrangean bounds, *Mathematical Programming* 39, 215-228.

[94] Guignard M., S. Kim (1987), Lagrangian decomposition for integer programming: theory and applications, *RAIRO-Recherche Opérationnelle* 21, 307-324.

[95] Hakimi S.L. (1971), Steiner's problem in graphs and its implications, *Networks* 1, 113-133.

[96] Hall L. (1996), Experience with a cutting plane algorithm for the capacitated spanning tree problem, *INFORMS Journal on Computing* 8, 219-234.

[97] Hanan M. (1966), On Steiner's problem with rectilinear distance, *SIAM Journal of Applied Mathematics* 14, 255-265.

[98] Handler G.Y., I. Zang (1980), A dual algorithm for the constrained shortest path problem, *Networks* 10, 293-310.

[99] Haouari M., J. Chaouachi (2006), A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem, *Computers and Operations Research* 33, 1274-1288.

[100] Haouari M., J. Chaouachi (2006), Lower and upper bounding strategies for the generalized minimum spanning tree problem, *European Journal of Operational Research* 171, 632-647.

[101] Haouari M., J. Chaouachi, M. Dror (2005), Solving the generalized minimum spanning tree problem by a branch-and-bound algorithm, *Journal of the Operational Research Society* 56, 382-389.

[102] Haouari M., S. Layeb, H.D. Sherali (2008), The prize collecting Steiner tree problem: models and Lagrangian dual optimization approaches, *Computational Optimization and Applications* 40, 13-39.

[103] Haouari M., S. Layeb, H.D. Sherali (2007), An exact solution approach for the prize collecting Steiner tree problem, (submitted).

[104] Haouari M., P. Pejax (1997), Plus court chemin avec dépendance horaire: résolution et application aux problèmes de tournées, *RAIRO-Operations Research* 31, 117-131.

[105] Held M., R.M. Karp (1971), The traveling-salesman problem and minimum spanning trees: part II , *Mathematical Programming* 1, 6-25.

[106] Held M., P. Wolfe, H.P. Crowder (1974), Validation of subgradient optimization, *Mathematical Programming* 6, 62-88

[107] Helvig C.S., G. Robins, A. Zelikovsky (2001), An improved approximation scheme for the group Steiner problem, *Networks* 37, 8-20.

[108] Hiriart-Urruty J.B, C. Lemaréchal (1993), *Convex analysis and minimization Algorithms*, Springer-Verlag.

[109] Holmberg K., J. Hellstrand (1998), Solving the uncapacitated network design problem by a Lagrangean heuristic and branch-and-bound, *Operations Research* 46, 247-259.

[110] Hu B., M. Leitner, G.R. Raidl (2008), Combining variable neighbourhood search with integer linear programming for the generalized minimum spanning tree problem, Journal of Heuristics 14, 473-499.

[111] Hwang F.K., D.S. Richards (1992), Steiner tree problems, *Networks* 22, 55-89.

[112] Hwang F.K., D.S. Richards, P. Winter (1992), The Steiner tree problem, *Annals of Discrete Mathematics* 53.

[113] Ihler E. (1994), The rectilinear class Steiner tree problem for intervals on two parallel lines, *Mathematical Programming* 63, 281 296.

[114] Ihler E., G. Reich, P. Widmayer (1999), Class Steiner trees and VLSI-design, *Discrete Applied Mathematics* 90, 173-194.

[115] Ikura Y., G.L. Nemhauser (1982), An efficient primal simplex algorithm for maximum weighted vertex packing on bipartite graphs, *Annals of Discrete Mathematics* 16, 149-168.

[116] Johnson D.S., M. Minkoff, S. Philips (2000), The prize collecting Steiner tree problem: theory and practice, *Proceedings of the 11th ACM-SIAM Symposium on Discrete Mathematics*, San Fransisco, CA, 760-769.

[117] Joksch H.C. (1966), The shortest route with constraints, *Journal of Mathematical Analysis and Applications* 14, 191-197.

[118] Jörnsten K., M. Nasberg (1986), A new Lagrangian relxation approach to the generalized assignement problem, *European Journal of Operational Research* 27, 313-323.

[119] Jörnsten K., A. Lokketangen (1997), Tabu search for weighted k-cardinality trees, *Asia-Pacific Journal of Operational Research* 14, 9-26.

[120] Jünger M., G. Reinelt, G. Rinaldi (1995), The travelling salesman problem, *Handbooks in Operational Research and Management Science* 7, 225-330.

[121] Kallehauge B., J. Larsen, O.B.G. Madsen (2006), Lagrangian duality applied to the vehicle routing problem with time windows, *Computers and Operations Research* 33, 1464-1487.

[122] Karger D.R., P.N. Klein, R.E. Tarjan (1995), A randomized linear-time algorithm to find minimum spanning trees, *Journal of the Association for Computing Machinery* 42, 321-328.

[123] Karnaugh M. (1976), A new class of algorithms for multipoint network optimization, *IEEE Transactions on Communications* 24, 500-505.

[124] Karp R.M. (1972), Reducibility among combinatorial problems, *Complexity of Computer Computations* 85–103.

[125] Kataoka S., N. Araki (2000), Upper and lower bounding procedures for minimum rooted k-subtree problem, *European Journal of Operational Research* 122, 561-569.

[126] Kelley J.E. (1960), The cutting-plane method for solving convex programs, *Journal of SIAM* 8, 703-712.

[127] Khuller S., A. Zhu (2002), The general Steiner tree-star problem, *Information Processing Letters* 84 , 215–220.

[128] Klein P.N., R. Ravi (1995), A nearly best-possible approximation algorithm for node-weighted Steiner trees, *Journal of Algorithms* 19,104-114.

[129] Knuth D.E. (1997), *The Art Of Computer Programming, Volume 1: Fundamental algorithms*, Addison-Wesley Publishing Compagny.

[130] Koch T., A. Martin (1998), Solving Steiner tree problems in graphs to optimality, *Networks* 32, 207-232.

[131] Koch T., A. Martin, Steinlib,

`http://elib.zib.de/steinlib/steinlib.php`.

[132] Kompella V.P., J.C. Pasquale, G.C. Polyzos (1993), Multicast routing for multimedia communication, *IEEE ACM* 1, 289-292.

[133] Kou L., G. Markovsky, L. Berman (1981), A fast algorithm for Steiner trees, *Acta Informatic* 15, 141-145.

[134] Kruskal J.B. (1956), On the shortest spanning subtree of a graph and travelling salesman problem, *Proceedings of American Mathematical Society* 7, 48-50.

[135] Langevin A., F. Soumis, J. Desrosiers (1990), Classification of traveling salesman problem formulations, *Operations Research Letters* 9, 127-132.

[136] Larson T., Z. Liu (1989), *A primal convergence result for dual subgradient optimization with application to multicommodity network flows*, Research reoprt S-581 83, Department of mathematics, Linkoping Institute of Technology, Linkoping, Sweden.

[137] Lawler E.L. (1976), *Combinatorial optimisation: networks and matroids*, Holt, Rinehart and Winston, New York.

[138] Lee Y., L. Lu, Y. Qiu, F. Glover (1994) , Strong formulations and cutting planes for designing digital data networks, *Telecommunication System* 2, 261-274.

[139] Lee Y., S. Chiu, J. Ryan (1996), A branch and cut algorithm for the Steiner tree/star problem, *INFORMS Journal on Computing* 8, 194–201.

[140] Legendre J.P., M. Minoux (1977), Une application de la notion de dualité en programmation en nombre entiers : séléction et affectation optimales d'une flotte d'avions, *Operations Research* 11, 201-222.

[141] Leggieria V., M. Haouari, S. Layeb, C. Triki (2007), The Steiner Tree problem with delays: a tight compact formulation and reduction procedures, (submitted).

[142] Lemaréchal C. (1975), An algorithm for minimizing convex functions, *Proceedings IFIP'75 Congress*, 552-556.

[143] Lemaréchal C. (2001), Lagrangian relaxation, *Lecture Notes in Computer Science* 2241, 112-156.

[144] Lim C., H.D. Sherali (2006), Convergence and computational analyses for some variable target value and subgradient deflection methods, *Computational Optimization and Applications* 34, 409-428.

[145] Lucena A. (1992), Steiner problem in graphs: Lagrangian relaxation and cutting planes, *COAL* 21, 2-7.

[146] Lucena A., J. Beasley (1998), A branch-and-cut algorithm for the Steiner problem in graphs, *Networks* 31, 3959.

[147] Lucena A., M.G.C. Resende (2004), Strong lower bounds for the prize collecting Steiner problem in graphs, *Discrete Applied Mathematics* 141, 277-294.

[148] Ljubic I., R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, M. Fischetti (2006), An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem, *Mathematical Programming* 10, 427-449.

[149] Maculan N., P. Michelon, A.E. Xavier (2000), The euclidean Steiner tree problem in IR^n: a mathematical programming formulation, *Annals of Operations Research* 96, 209-220.

[150] Maffioli F. (1981), Complexity of optimum undirected tree problems: a survey of recent results, *CISM Courses and Lectures* 266, 107-128.

[151] Magnanti T.L., L.A. Wolsey (1995), Optimal trees, in *Handbooks in Operational Research and Management Science* 7, 503-615.

[152] Malik K., G. Yu (1993), A branch-and-bound algorithm for the capacitated minimum spanning tree problem, *Networks* 23, 525-532.

[153] Mamadi D., J. Plesnik (1993), An integer programming formulation of the Steiner problem in graphs, *Methods and Models of Operations Research* 37, 107-111.

[154] Marsten R.E., W.W. Hogan, J.W. Blankenship (1975), The Boxstep method for large-scale optimization, *Operations Research* 23, 389-405.

[155] S.L. Martins, M.G.C.Resende, C.C. Ribeiro, P.M. Pardalos (2000), A parallel GRASP for the steiner tree problem in graphs using a hybrid local search strategy, *Journal of Global Optimization* 17, 267-283.

[156] Melkonian V. (2007), New primal-dual algorithms for Steiner tree problems, *Computers & Operations Research* 34, 2147-2167.

[157] Michelon P., N. Maculan (1988), Lagrangian decomposition for integer non-linear programming, *Mathematical Programming* 52, 303-314.

[158] Miller C.E., A.W. Tucker, R.A. Zemlin (1960), Integer programming formulations and traveling salesman problems, *Journal of the Association for Computing Machinery* 7, 326-329.

[159] Minoux M. (1975), Plus court chemin avec contraintes: algorithmes et applications, *Annales des télécommunications* 30, 383-394.

[160] Molnár M. (1998), Construction d'arbres de diffusion multicast basée sur une modification de l'heuristique de Kruskal, *rapport de recherche IRISA* N°1214.

[161] Myung Y.S., C.H. Lee, D.W. Tcha (1995), On the generalized minimum spanning tree problem, *Networks* 26, 231-241.

[162] Nemhauser G.L., L.A. Wolsey (1999), *Integer and combinatorial optimization*, Second edition, Wiley Interscience, New York.

[163] Nešetřil J., E. Miková, H. Nešetřilová (2001),Otakar Borůvka on minimum spanning tree problem: translation of both the 1926 papers, comments, history, *Discrete Mathemathics* 233, 3-36.

[164] Noronha C., F. Tobagi (1994), Optimum routing of multicast streams, *Proceedings of IEEE INFOCOM'94* 2 , 865-873.

[165] Padberg M., T. Sung (1992), An analytical comparison of different formulations of the traveling salesman problem, *Mathematical Programming* 52, 315-357.

[166] Papadimitriou C. (1978), The complexity of the capacitated tree problem, *Networks* 8, 217-230.

[167] Papadimitriou C. (1994), *Computational complexity*, Addison-Wesley, Reading.

[168] Pierce A.R. (1975), Bibliography on algorithms for shortest path, shortest spanning tree and related circuit routing problems (1956-1974), *Networks* 5,129-149.

[169] Polyak B.T. (1967), A general method of solving extremum problems, *Soviet Mathematics Doklady* 8, 593-597.

[170] Polyak B.T. (1969), Minimization of unsmooth functionals, *U.S.S.R. Computational Mathematics and Mathematical Physics* 9, 14-29.

[171] Polzin T., S. Vahdati Daneshmand (2001), A comparison of Steiner tree relaxations, *Discrete Applied Mathematics* 112, 241-261.

[172] Pop P.C. (2007), On the prize-collecting generalized minimum spanning tree problem, *Annals of Operations Research* 150, 193-204.

[173] Pop P.C. (2002), *The generalized minimum spanning tree problem*, PhD thesis, University of Twente, The Netherlands.

[174] Pop P.C., W. Kern, G.J. Still (2001), *An approximation algorithm for the generalized minimum spanning tree problem with bounded cluster size*, Faculty of Mathematical Sciences , University of Twente, University for Technical and Social Sciences, Memorandum N°1577.

[175] Prim R.C. (1957), Shortest connection networks and some generalisations, *Bell System Technical Journal* 36, 1389-1401.

[176] Prins C. (1994), *Algorithmes de graphes avec programmes en Pascal*, EYROLLES.

[177] Reich G., P. Widmayer (1990), Beyond Steiner's problem: a VLSI oriented generalization, *Lecture Notes in Computer Science* 411, 196-210.

[178] Ribeiro C.C. (1983), *Algorithmes de plus courts chemins avec contraintes: étude théorique, implémantation et parallélisation*, Thèse de Docteur Ingénieur.

[179] Ribeiro C.C., M.C. De Souza (2000), Tabu search for the Steiner problem in graphs, *Networks* 36, 138–146.

[180] Robins G., A. Zelikovsky (2005), Tighter bounds for graph Steiner tree approximation, *SIAM Journal of Discrete Mathemathics* 19, 122-134.

[181] Ross G.T., R.M. Soland (1975), A Branch and bound algorithm for the generalized assignment problem, *Mathematical Programming* 8, 91-103.

[182] Salazar J.J (2000), A note on the generalized Steiner tree polytope, *Discrete Applied Mathematics* 100, 137-144.

[183] Sharaiha, Y.M., M. Gendreau, G. Laporte, I.H. Osman (1997), A tabu search algorithm for the capacitated minimum spanning tree problem, *Networks* 29, 161-171.

[184] Segev A. (1987), The node-weighted Steiner tree problem, *Networks* 17, 1-17.

[185] Takahashi H., A. Matsuyama (1981), An approximate solution for the Steiner problem in graphs, *Mathematical Japonica* 24, 573-577.

[186] Tseng S.Y., Y.M. Huang, C.C. Lin (2006), Genetic algorithm for delayand degree-constrained multimedia broadcasting on overlay networks, *Computer Communications* 29, 3625-3632.

[187] Shapiro J.F. (1979), A survey of Lagrangian techniques for discrete optimization, *Annals of Discrete Mathematics* 5, 113-38.

[188] Shepardson F., R. Marsten (1980), A Lagrangian relaxation algorithm for the two duty period scheduling problem, *Management science* 26, 274-281.

[189] Sherali H.D., W.P. Adams, *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*, Kluwer Academic, 1999.

[190] Sherali H.D., W.P. Adams (1990), A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems, *SIAM Journal of Discrete Mathematics* 3, 411-430.

[191] Sherali H.D., W.P. Adams (1994), A hierarchy of relaxations and convex hull characteristics for mixed-integer zero-one programming problems, *Discrete Applied Mathematics* 52, 83-106.

[192] Sherali H.D., I. Al-Loughani (1999), Solving euclidean distance multifacility location problems using conjuate subgradient and line-search methods, *Computational Optimization and Applications* 14, 275-291.

[193] Sherali H.D., G. Choi (1996), Recovery of primal solutions when using subgradient optimization methods to solve lagrangien duals of linear programs, *Operations Research Letters* 19, 509-521.

[194] Sherali H.D., G. Choi, Z. Ansari (2001), Limited memory space dilatation and reduction algorithms, *Computational Optimization and Applications* 19, 55-77.

[195] Sherali H.D., G. Choi, C.H. Tuncbilek (2000), A variable target value method for nondifferentiable optimization, *Operations Research Letters* 26, 1-8.

[196] Sherali H.D., P.J. Driscoll (2002), On tightening the relaxations of Miller-Tucker-Zemlin formulations for asymmetric traveling salesman problems, *Operations Research* 50, 656-669.

[197] Sherali H.D., S.C. Sarin, P. Tsai (2006), A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints, *Discrete Optimization* 3, 20-32.

[198] Sherali H.D., O. Ulular (1989), A primal-dual conjugate subgradient algorithm for specially structured linear and convex programming problems, *Applied Mathematics and Optimization* 20, 193-221.

[199] Sherali H.D., O. Ulular (1990), Conjugate gradient methods using quasi-Newton updates with inexact line searches, *Journal of Mathematical Analysis and Applications* 150, 359-377.

[200] Shor N.Z. (1985), *Minimization methods for nondifferentiable functions*, Springer-Verlag, Berlin.

[201] Stanojevic D., B. Golden, S. Raghavan (2004), The prize-collecting generalized minimum spanning tree problem, 7^{th} *INFORMS Telleco-mucations Conference*, Boca Raton, Florida.

[202] Véjar A.C., H.B. Azlàn (2004), Performance analysis of algorithms for the Steiner problem in directed networks, *Electronic Notes in Discrete Mathematics* 18, 67-72.

[203] Voβ S. (1999), The Steiner tree problem with hop constraints, *Annals of Operations Research* 86, 321-345.

[204] Wah W.B., Z. Wu (2004), The theory of discrete lagrange multipliers for nonlinear discrete optimization, *Lecture Notes in Computer Science* 1713, 28-42.

[205] Warme D.M (1997), *A new exact algorithm for rectilinear Steiner minimal trees*, Technical report, System Simulation Solutions, Inc., Alexandria, USA.

[206] Weng J.F. (1985), Generalized Steiner problem and hexagonal coordinate system, *Acta Mathematicae Applicatae Sinica* 8, 383-397.

[207] Winter P. (1987), Steiner problem in networks: a survey, *Networks* 17, 129-167.

[208] Wilson R.J., J.J. Watkins (1990), *Graphs : an introductory approach*, Jhon Wilsey & Sons, Inc.

[209] Wong R.T. (1984), A dual ascent approach for Steiner tree problems on a directed graph, *Mathematical Programming* 28, 271-287.

[210] Xu J., S.Y. Chiu, F. Glover (1996), Using tabu search to solve the Steiner tree-star problem in telecommunications network design, *Telecommunication Systems* 6, 117-125.

[211] Yang B. (2006), A hybrid evolutionary algorithm for the euclidean Steiner tree problem using local searches, *Lecture Notes in Computer Science* 4251, 60-67.

[212] Yang B. (2005), An evolution algorithm for the rectilinear Steiner tree problem, *Book Series Lecture Notes in Computer Science* 3483, 241-249.

[213] Yang B., P. Gillard (2000), The class Steiner minimal tree problem: a lower bound and test problem generation, *Acta Informatica* 37, 193-211.

[214] Zachariasen M., A. Rohe (2003), Rectilinear group Steiner trees and applications in VLSI design, *Mathematical Programming* 94, 407-433.