The multicommodity flow blocker problem : polyhedral analysis and algorithms

Isma Bentoumi

PSL UNIVERSITÉ PARIS

HUAWEI

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'Université Paris Dauphine

The multicommodity flow blocker problem: Polyhedral analysis and algorithms

Soutenue par
Isma BENTOUMI
Le 08 juillet 2024

École doctorale n°543
SDOSE

Spécialité
Informatique

Composition du jury :

Ivana LJUBIC ESSEC Business School de Paris	*Rapporteur*
Alain QUILLIOT Université Clermont-Auvergne	*Rapporteur*
Roland GRAPPE Université Paris Dauphine	*Examinateur*
Nancy PERROT Orange Labs France	*Examinatrice*
Fatiha BENDALI Université Clermont-Auvergne	*Examinatrice*
Ibrahima DIARRASSOUBA Université Le Havre Normandie	*Membre invité*
Ali Ridha MAHJOUB Université Paris Dauphine	*Directeur de thèse*
Fabio FURINI Sapienza Université de Rome	*Co-directeur*
Sébastien MARTIN Huawei Technologies France	*Co-encadrant*

Dauphine | PSL UNIVERSITÉ PARIS

Contents

Remerciements

Au travers de ces quelques lignes, je souhaite remercier toutes les personnes qui, par leur soutien, leurs conseils et leurs encouragements, ont contribué à l'aboutissement de cette thèse.

Je tiens tout d'abord à exprimer ma profonde gratitude à mes directeurs de thèse, Monsieur Ridha Mahjoub, Monsieur Fabio Furini et Monsieur Sébastien Martin, pour m'avoir offert l'opportunité de réaliser cette thèse sous leur direction.

Je tiens à exprimer mes sincères remerciements à Monsieur Ali Ridha Mahjoub, professeur émérite à l'Université Paris Dauphine PSL, pour avoir partagé avec moi son savoir et son expérience précieuse.

Je suis extrêmement reconnaissante envers Monsieur Fabio Furini, professeur à Sapienza, Université de Rome. J'aimerai le remercier pour les (très) longues sessions de travail que nous avons partagé ainsi que pour sa disponibilité et son engagement constant tout au long de la thèse. Je le remercie pour m'avoir enseigné la rigueur, la réflexion, la rédaction. J'ai beaucoup appris à ses côtés, tant au niveau scientifique qu'au niveau personnel. *Fabio, grazie mille.*

Je remercie infiniment Monsieur Sébastien Martin, ingénieur de recherche à Huawei Paris Research Center, pour m'avoir permis de réaliser ce projet de thèse en collaboration avec Huawei. Son soutien, sa bienveillance et son aide précieuse ont été déterminants pour l'aboutissement de mon travail. Je le remercie pour ses conseils avisés mais aussi pour m'avoir toujours incité à repousser les limites de mes connaissances, notamment en me challengeant sur des sujets divers de l'optimisation combinatoire.

Je souhaiterai remercier Madame Ivana Ljubic, professeur à l'ESSEC Business School of Paris, pour m'avoir fait l'honneur de rapporter ma thèse. Je la remercie pour ses conseils ainsi que pour ses corrections méticuleuses.

Je remercie également Monsieur Alain Quilliot, professeur émérite à l'Université de Clermont-Ferrand, pour avoir accepté de rapporter ma thèse, ainsi que pour l'intérêt qu'il a porté à mon travail et ses suggestions pertinentes.

Ma gratitude va également à Madame Fatiha Bendali, professeur à l'Université de Clermont-Ferrand, d'avoir accepté de présider le jury de ma thèse.

Enfin, je remercie Monsieur Roland Grappe, professeur à l'Université Paris Dauphine PSL, Madame Nancy Perrot, ingénieur de recherche à Orange Labs, ainsi que Monsieur Ibrahima Diarrassouba, Maitre de conférences à

l'Université du Havre, pour m'avoir fait l'honneur de faire partie de mon jury de thèse et pour leurs interventions pertinentes durant la soutenance.

Je tiens également à remercier chaleureusement tous les membres de l'équipe RO à Huawei Paris Research Center. Je les remercie pour leur soutien et leurs conseils durant les trois premières années de ma thèse. Leur expertise fut très enrichissante. Je remercie plus particulièrement Youcef pour nos discussions scientifiques approfondies autour de l'analyse polyèdrale ainsi que Pierre et Anne pour des sujets un peu plus légers.

Je remercie également tous les membres du LAMSADE. En particulier, les doctorants des bureaux C603bis, C602 et C605 pour leur accueil et les bons moments passés en leur compagnie. Merci notamment à Charles pour nos discussions enrichissantes autour de l'optimisation et à la team de la cité universitaire, Sébastien, Manolis, Virginia et Sofia pour les longues sessions de travail dans la bibliothèque (week-ends inclus) et pour les tout aussi longues pauses passées sur la pelouse.

Enfin, je tiens à remercier tous mes amis, d'Alger à Paris Dauphine, en passant par Paris Diderot et Polytech, pour avoir rendu toutes ces années d'études si agréables.

Pour finir, mes remerciements vont à ma famille, sans qui l'aboutissement de cette thèse ainsi que chaque étape qui l'ont précédé, n'auraient jamais pu être possible. Merci du fond du cœur à mon père et ma mère pour avoir été des modèles aussi inspirants, pour leur soutien inconditionnel et pour m'avoir transmis les valeurs essentielles du travail et de la persévérance. Un immense merci à mes sœurs, Tinhinane, Kahina et Ines, ainsi qu'à mon beau-frère Seif, pour leur soutien, leur aide, leurs conseils et pour leur capacité à toujours trouver les mots justes pour m'encourager.

Abstract

Telecommunication networks are complex systems in which hard combinatorial optimization problems must be solved. With the increasing demand, telecommunication networks have to be efficient, especially in terms of time lag and failure tolerance. In this context, this thesis will focus on the resilience analysis of a network. More precisely, the primary goal of this study is to determine the maximum number of anomalies that may occur in the network while guaranteeing its functionality depending on a crucial property that needs to be maintained.

This challenge belongs to a class of optimization problems called network flow blocker problems. The initial purpose of this work is to focus on multicommodity flow problems arising in telecommunication networks with complex demand satisfaction constraints. We are interested in demands, also called commodities, having different shapes of traffic and predicted with machine learning methods. The associated network flow blocker problem is called multicommodity flow blocker problem (MCFBP). To tackle this challenge, we use combinatorial optimization tools, delving into various approaches including bilevel techniques that exploit the bilevel nature of network flow blocker problems, a polyhedral approach and branching algorithms.

We first focus on the maximum flow blocker problem (MFBP), which is a particular case of the MCFBP. To solve the MFBP, we propose several Integer Linear Programming (ILP) formulations and a technique derived from a theoretical result that establishes a structural link between the MFBP and another problem existing in the literature. We then expand our work to address the blocker notion of the multicommodity flow problem, covering several variants. For this problem, we introduce an ILP formulation featuring an exponential number of constraints and solved using a tailored branch-and-cut algorithm. In addition, we enhance the model's robustness by studying its polyhedra. Performance of the exact methods proposed to solve the MFBP and the MCFBP are evaluated through an extensive computational campaign involving a set of synthetic and real-world instances.

Keywords : Combinatorial optimization, maximum flow, multicommodity flow, blocker, interdiction, bilevel optimization

Résumé

Les réseaux de télécommunication sont des systèmes complexes dans lesquels s'inscrivent des problèmes d'optimisation combinatoire difficiles. Face à une demande croissante, l'efficacité de ces réseaux devient cruciale, notamment en termes de délai et de résilience face aux anomalies. Cette thèse se concentre sur l'analyse de la résilience des réseaux, avec pour objectif principal de déterminer le nombre maximal d'anomalies que le réseau peut supporter tout en maintenant sa fonctionnalité selon des critères spécifiques.

Ce défi relève d'une classe de problèmes d'optimisation connus sous le nom de problèmes de bloqueur sur les flots. Notre travail se focalise particulièrement sur les problèmes de multiflots rencontrés dans les réseaux de télécommunication, caractérisés par des contraintes complexes de satisfaction des demandes. Nous nous intéressons aux demandes ayant différentes formes de trafic et prédites avec des méthodes d'apprentissage automatique. Le problème étudié est appelé problème de bloqueur sur les multiflots. Pour répondre à cette problématique, nous utilisons des outils d'optimisation combinatoire, explorant diverses approches, y compris les techniques bi-niveau, l'approche polyèdrale et les algorithmes de branchement.

Nous abordons initialement le problème du bloqueur sur le flot maximal. Pour le résoudre, nous proposons plusieurs formulations de programmation linéaire entière, ainsi qu'une technique dérivée d'un résultat théorique établissant un lien structurel avec un problème existant dans la littérature. Par la suite, nous étendons notre travail pour traiter la notion de bloqueur sur les multiflots, en couvrant plusieurs variantes. Pour ce faire, nous introduisons une formulation avec une famille exponentielle de contraintes, résolue à l'aide d'un algorithme de branchement et de coupes. De plus, nous développons une approche polyèdrale pour renforcer la robustesse du modèle. Les performances des méthodes exactes proposées pour résoudre le deux problèmes décrits sont évaluées à travers une étude expérimentale approfondie.

Mots clés : Optimisation combinatoire, flot maximum, multiflots, bloqueur, interdiction, optimisation biniveau

Résumé long

Introduction

L'industrie des Télécommunications est un secteur en pleine expansion, poussé par une demande croissante et confronté à l'émergence de nouveaux défis. L'une des dernières innovations est la technologie 5G, visant à garantir un niveau élevé de fiabilité de la communication. Cependant, la question des pertes de signal dans la transmission de trafic reste une préoccupation critique qui nécessite une attention particulière. Ainsi, les réseaux de télécommunications modernes constituent des systèmes complexes, nécessitant la résolution efficace de problèmes d'optimisation combinatoire difficiles dans des délais de calcul courts. Ces problèmes relèvent de la catégorie des problèmes d'optimisation dans les réseaux. Parmi les plus connus de cette catégorie figurent le problème de flot maximum et le problème de flot multi-commodités, qui constituent l'objectif principal de cette thèse.

Les réseaux de télécommunications se composent de dispositifs de communication interconnectés et d'infrastructures conçues pour faciliter le routage et la transmission d'informations. Ces réseaux peuvent être modélisés comme des graphes, où les sommets représentent des routeurs reliés par des arcs, symbolisant des liens de transmission. Les liens de transmission, cruciaux pour le processus de routage et facilitant le flux de données, incluent des informations supplémentaires, telles que la capacité des arcs, indiquant le volume de données maximal pouvant être transmis à travers chaque arc. Le routage au sein de ces réseaux implique la transmission de données d'une source à une destination, correspondant généralement à une demande à satisfaire, aussi appelé commodité. Dans des situations réelles, ces réseaux présentent des dimensions étendues et peuvent être accompagnés d'un volume significatif de demandes. Néanmoins, souvent, les réseaux de télécommunications sont confrontés à des anomalies et des pannes.

Différents types d'anomalies peuvent survenir dans un réseau de télécommunications et affecter ses performances. La congestion du réseau, un événement courant, se produit lorsque la demande dépasse la capacité du réseau, entraînant une réduction de la vitesse de transfert de données. Les pannes d'équipement, qu'elles soient dues à des dysfonctionnements matériels ou à des facteurs environnementaux, peuvent perturber la transmission du signal et compromettre la fiabilité du réseau. Les cyberattaques et les violations de

sécurité introduisent des anomalies délibérées, compromettant la confidentialité des données et l'intégrité du réseau. Les interférences de signal provenant de sources externes, telles que les radiations électromagnétiques ou des obstacles physiques, peuvent entraîner des pertes de signal. De plus, des bugs logiciels et des erreurs de configuration peuvent déclencher des comportements inattendus et provoquer des interruptions de service. La complexité de ces anomalies met en évidence la nécessité d'intégrer des systèmes de détection d'anomalies capables d'évaluer la capacité du réseau à y faire face. Ainsi, des stratégies robustes peuvent être élaborées pour renforcer la résilience du réseau, réduire les pannes et assurer une communication fluide. Aussi, les opérateurs de réseaux sont souvent confrontés à la question d'identifier la partie la plus vitale (également appelée la plus vulnérable ou la plus critique) du réseau, qui correspond à un sous-ensemble de sommets (ou d'arcs), dont le dysfonctionnement empêche le bon fonctionnement du réseau dans son ensemble.

Dans ce contexte, ce travail se concentre sur l'analyse de la résilience d'un réseau. Plus précisément, l'objectif principal de cette étude est de déterminer le nombre maximal d'anomalies pouvant survenir dans le réseau tout en garantissant son bon fonctionnement en fonction d'une propriété cruciale qui doit être maintenue (ou atteinte). Cette propriété peut englober divers aspects du réseau, incluant des contraintes telles que le routage d'un volume spécifique de données ou le respect d'un budget de routage. Ce défi appartient à une classe de problèmes d'optimisation appelée problèmes de bloqueur dans les réseaux.

D'un point de vue d'optimisation combinatoire, les anomalies peuvent être modélisées avec la notion de bloqueur. Celle-ci permet de déterminer le nombre minimum de perturbations nécessaires pour détruire une structure spécifique. Ici, les perturbations représentent les anomalies, et la structure spécifique correspond à un réseau. Plus formellement, dans un réseau où chaque arc (ou sommet) est associé à un coût de suppression, un problème de bloqueur dans le réseau vise à trouver un ensemble d'arcs (ou de sommets) à supprimer du graphe, avec un coût minimal, de sorte que le réseau ne soit plus fonctionnel, c'est-à-dire qu'il ne satisfasse plus une propriété spécifique. Les arcs (ou sommets) supprimés représentent les anomalies pouvant survenir dans le réseau et détruire les liens de transmission (ou les routeurs). Par conséquent, la résolution d'un problème de bloqueur dans un réseau fournit le nombre maximal d'anomalies que le réseau peut supporter tout en maintenant sa fonctionnalité. Plus précisément, le nombre maximal d'anomalies est équivalent à la valeur de la solution du bloqueur moins un.

L'objectif initial de ce travail est de se concentrer sur les problèmes de flot multi-commodités qui se posent dans les réseaux de télécommunications avec des contraintes complexes de satisfaction de la demande. Nous nous intéressons aux demandes, également appelées commodités, présentant différentes formes de trafic et prédites par des méthodes d'apprentissage automatique. Le problème de bloqueur associé est appelé problème de bloqueur de flot

multi-commodités (MCFBP), où le réseau est représenté par un problème de flot multi-commodités sur lequel un problème de bloqueur est appliqué. Dans ce contexte, nous supposons qu'un réseau est opérationnel s'il existe un flot multi-commodités dans le réseau satisfaisant une propriété donnée. Le problème de bloqueur de flot multi-commodités évalue le poids maximal des anomalies qui peuvent survenir dans le réseau sans qu'il échoue. Ainsi, une solution au problème de bloqueur de flot multi-commodités est un indicateur définissant la tolérance d'un réseau face aux anomalies, ce qui constitue un élément central pour améliorer l'efficacité du routage. Nous étudions également un cas particulier du MCFBP, qui est le problème de bloqueur de flot maximum (MFBP). Dans le MFBP, le réseau est représenté par un problème de flot maximum dont l'objectif est d'envoyer une quantité maximale de données d'une source unique à une destination unique. Pour aborder ces problèmes, nous utilisons des outils d'optimisation combinatoire, en explorant diverses approches, notamment des techniques bi-niveaux qui exploitent la nature bi-niveau des problèmes de bloqueur dans les réseaux, l'approche polyèdrale initiée par Edmonds [1965] et les algorithmes de branchement.

Pour le MFBP, nous présentons plusieurs formulations de Programmation Linéaire en Nombres Entiers (PLNE) conçues pour la première fois afin de traiter ce problème. Les premières formulations, comportant un nombre exponentiel de contraintes, sont résolues à l'aide d'algorithmes de Branch-and-Cut adaptés. Un autre modèle PLNE, avec un nombre polynomial de variables et de contraintes, est résolu à l'aide d'un solveur PLNE. Cette dernière formulation établit une connexion structurelle entre le problème de bloqueur de flot maximum et un autre problème proche existant dans la littérature, appelé problème d'interdiction de flot maximum. Cette relation introduit une approche novatrice permettant d'obtenir des solutions pour chaque problème à partir de l'autre. Les méthodes exactes proposées sont comparées à travers une étude expérimentale poussée impliquant un ensemble d'instances synthétiques et réelles, visant à évaluer leurs performances.

Nous étendons ensuite notre recherche pour aborder la variante du problème de bloqueur de flot multi-commodités. À cet effet, nous introduisons une formulation comprenant une famille exponentielle de contraintes. Celle-ci est résolue à l'aide d'un algorithme de Branch-and-Cut sur mesure. De plus, nous renforçons la robustesse du modèle en développant une approche polyédrale. Plusieurs variantes du problème ont été explorées, y compris une concernant une instance spécifique du problème où le flot est dit discret, c'est-à-dire qu'il est contraint d'être routé via un seul chemin pour chaque commodité. Une autre variante est la variante de bloqueur sur les sommets, qui consiste à supprimer des sommets du graphe à la place des arcs. Tous les algorithmes exacts conçus pour le problème de bloqueur de flot multi-commodités et ses variantes sont évalués expérimentalement sur un ensemble d'instances synthétiques et réelles.

Cette thèse explore également plusieurs applications étroitement liées aux réseaux de télécommunications. En outre, dans le cadre de notre étude sur

l'analyse de la résilience des réseaux, nous avons approfondi une application spécifique liée à la surveillance du réseau. Cela implique de déterminer le placement optimal de structures de données, appelées sketches, pour surveiller efficacement le trafic.

Preliminaries and State-of-the-art

Le premier chapitre est dédié à la présentation des notions préliminaires concernant l'optimisation combinatoire, l'analyse polyédrale, les problèmes d'optimisation dans les réseaux ainsi que l'optimisation bi-niveau. Précisément, nous fournissons un aperçu des méthodes bien connues utilisées en optimisation combinatoire, telles que les algorithmes de Branch-and-Bound, Branch-and-Cut et Branch-and-Price. Après avoir introduit quelques éléments fondamentaux de théorie des graphes, nous présentons des notations relatives aux problèmes de flot dans les réseaux. Ces notations seront utilisées tout au long du document. Nous donnons ensuite un aperçu de la programmation bi-niveau avec un état de l'art sur certains problèmes bi-niveau, tels que le problème d'interdiction et le problème de bloqueur.

Network flow blocker problems and their applications in telecommunication networks

Dans ce chapitre, nous examinons en profondeur l'architecture des réseaux de télécommunications et les anomalies auxquelles ils peuvent être confrontés. Nous explorons ensuite les applications pratiques liées aux problèmes de bloqueur dans les réseaux en nous concentrons principalement sur la détection des anomalies. Plus précisément, nous démontrons une corrélation directe entre le nombre maximal d'anomalies pouvant survenir dans le réseau et la solution d'un problème de bloqueur dans ce réseau. Par la suite, nous nous plongeons dans le domaine de la surveillance des réseaux, et plus particulièrement dans le placement de structures de données de surveillance appelées sketches. Dans ce contexte, nous examinons le problème d'attribution de sketches, un problème d'optimisation visant à placer stratégiquement les sketches au sein d'un réseau. Pour traiter ce problème, nous proposons une formulation bi-niveau qui sera ensuite résolue à l'aide d'un algorithme glouton.

Comme expliqué dans ce chapitre, les problématiques abordées dans cette thèse présentent une importance particulière pour les opérateurs de réseau, qui cherchent à optimiser l'utilisation des ressources, garantir une connectivité fiable et offrir un service de haute qualité à leurs utilisateurs.

The maximum flow blocker problem: Formulations and algorithms

Dans ce chapitre, nous étudions le problème de bloqueur de flot maximum (MFBP).

Etant donné un graphe constitué d'un ensemble de nœuds et d'un ensemble d'arcs, chaque arc possède une capacité ainsi qu'un coût de suppression. Le problème de bloqueur de flot maximum (MFBP) consiste à déterminer un ensemble d'arcs dont le coût de suppression est minimal, de manière à ce que le flot maximal restant dans le graphe soit inférieur à un seuil prédéfini, appelé flot cible.

Ce problème peut être modélisé comme un problème d'optimisation biniveau. Le problème de premier niveau, également appelé leader, est un problème de bloqueur qui a pour but de retirer des arcs dans le graphe. Le problème de second niveau, également appelé follower, est un problème de flot maximum.

En utilisant la formulation bi-niveau naturelle du problème, nous avons introduit quatre nouvelles formulations de programmation linéaire entière afin de résoudre le MFBP.

Les trois premières formulations, qui impliquent uniquement les variables naturelles associées aux arcs, sont résolues à l'aide d'algorithmes de séparation et de coupes (Branch-and-cut).

La quatrième et dernière formulation est un modèle compacte résolue à l'aide d'un solveur ILP. Celle-ci comporte un nombre polynomiale de variables et de contraintes.

Les formulations naturelles (non compactes) présentent deux familles exponentielles de contraintes. La première, dont la séparation est polynomiale pour les solutions fractionnaires et entières, est appelée coupes de Benders. Pour la seconde famille d'inégalités, appelé inégalité de flot cible, la séparation des solutions fractionnaires est NP-difficile. En utilisant ces résultats, nous avons développé un algorithme Branch-and-Cut amélioré avec des coupes de Benders et des inégalités de flot cible.

En utilisant la formulation compacte, nous avons établi la première connexion entre le problème de bloqueur de flot maximum et le problème d'interdiction de flot maximum.

Etant donné un graphe constitué d'un ensemble de nœuds et d'un ensemble d'arcs, chaque arc possède une capacité ainsi qu'un coût de suppression. Le problème d'interdiction de flot maximum (MFIP) consiste à trouver un ensemble d'arcs à retirer du graphe, de manière à minimiser le flot maximal restant et tel que le cout total de suppression des arcs ne dépassent pas un budget prédéfini.

Nous avons démontré que les solutions d'un MFBP peuvent être obtenues à partir des solutions d'un MFIP, et réciproquement. Cette propriété a été

étendue à plusieurs variantes du problème, notamment la variante continue du bloqueur et la variante de bloqueur sur les sommets, qui consiste à retirer des sommets du graphe plutôt que des arcs.

Par ailleurs, nous avons réalisé une étude pour comparer la robustesse des relaxations linéaires des formulations proposées.

The maximum flow blocker problem: Computational experiments

Dans le chapitre précédent, nous avons présenté et comparé théoriquement plusieurs formulations pour aborder le MFBP. Nous avons d'abord conçu trois formulations naturelles. Ces formulations présentent un nombre exponentiel de contraintes et sont résolues par des algorithmes de Branch-and-Cut. Nous avons ensuite présenté une autre formulation ayant un nombre polynomial de variables et de contraintes, désignée comme formulation compacte. Celle-ci est résolue à l'aide d'un solveur ILP.

Dans ce chapitre, une analyse computationnelle approfondie a été réalisée pour évaluer la performance des formulations conçues pour le MFBP.

Nos tests ont montré que les algorithmes proposés dans cette thèse surpassent de manière significative l'utilisation directe d'un solveur bi-niveau, qui représente la technique actuelle existant dans la littérature pour traiter le MFBP. De plus, la formulation compacte s'est avérée être la plus efficace, surpassant les formulations naturelles et démontrant sa capacité à gérer des instances de grande taille de manière efficace dans un temps raisonnable.

The multicommodity flow blocker problem: Formulations and polyhedral analysis

Dans ce chapitre, nous approfondissons le problème de bloqueur de flot multi-commodités, en explorant à la fois ses variantes continues (MCFBP) et discrètes (UMCFBP). La variante continue concerne le cas où le flot de chaque commodité peut être réparti sur plusieurs chemins, tandis que dans la variante discrète, le flot de chaque commodité est contraint de passer entièrement par un seul chemin reliant la source à la destination.

Considérons un graphe constitué d'un ensemble de nœuds et d'arcs. Chaque arc est défini par une capacité, un coût de flot correspondant au routage d'une unité de flot à travers cet arc, ainsi qu'un coût de suppression. Nous prenons également en compte un ensemble de commodités, chacune étant caractérisée par un nœud source, un nœud destination, une bande passante représentant la quantité de flot à acheminer entre ces deux nœuds, et une récompense associée au routage d'une unité de flot. L'objectif du problème de second niveau, qui est un problème de flot multi-commodités, consiste à satisfaire un

maximum de commodités en acheminant une quantité de flot proche de la bande passante, afin de maximiser le gain total, calculé comme la somme des récompenses moins le coût de routage. Le problème de premier niveau est un problème de bloqueur dont l'objectif est de réduire le gain du flot multi-commodités restant dans le graphe après suppression des arcs, jusqu'à ce qu'il soit inférieur à seuil prédéfini.

Avant de résoudre ce problème, nous menons une analyse complète de la complexité du problème associée aux deux variantes, la variante continue et la variante discrète. Nous établissons par ailleurs une corrélation entre leurs solutions.

L'approche de résolution que nous proposons se base d'abord sur une formulation biniveau pour aborder ces défis, ainsi qu'une reformulation à un seul niveau. Nous présentons ensuite une formulation plus générale de programmation linéaire en nombres entiers (ILP) avec un nombre exponentiel de contraintes. Afin d'améliorer la robustesse du modèle, nous introduisons une étude polyédrale, visant à décrire les polyèdres des solutions pour chaque problème et à identifier les conditions nécessaires et suffisantes pour que les inégalités définissent des facettes.

Pour de futures recherches, l'exploitation de la formulation biniveau et de sa reformulation pourrait s'avérer bénéfique pour améliorer les modèles proposés en y intégrant de nouvelles inégalités valides.

A Branch-and-cut algorithm for the multicommodity flow blocker problem

Ce chapitre présente des algorithmes conçus pour résoudre les problèmes MCFBP et UMCFBP, en se focalisant sur des méthodes de type Branch-and-Cut. Ces algorithmes sont décrits en détail et soumis à une évaluation expérimentale rigoureuse.

Pour chaque variante, nous explorons en profondeur le problème de séparation lié à la famille exponentielle de contraintes proposée, et introduisons diverses approches pour séparer ces inégalités. Pour les solutions fractionnaires, où le problème de séparation est NP-complet, nous proposons une heuristique basée sur des calculs successifs de plus courts chemins.

De plus, pour la variante discrète du problème (UMCFBP), nous développons un algorithme reposant sur la génération de colonnes afin de séparer les points entiers infaisables.

Pour évaluer l'efficacité des différentes méthodes proposées et la performance de l'algorithme Branch-and-Cut, nous réalisons une étude computationnelle approfondie sur un ensemble d'instances synthétiques et réelles. Cette étude met en évidence l'efficacité des heuristiques, particulièrement sur des graphes de grande taille.

The multicommodity flow blocker problem: vertex variant

Dans les chapitres précédents, notre travail s'est concentré sur la variante arc du problème de bloqueur sur le flot maximum et sur le flot multi-commodités. Pour ces deux problèmes, l'objectif est de détruire des arcs du graphe afin de réduire respectivement le flot maximum et le flot multi-commodités.

Ce chapitre introduit une nouvelle variante du problème de bloqueur, qui consiste à supprimer des sommets plutôt que des arcs. Pour cela, nous nous intéressons plus spécifiquement au flot multi-commodités discret. Ce problème est appelé V-UMCFBP.

Après avoir décrit formellement le V-UMCFBP, nous établissons un lien entre les solutions de la variante du problème de bloqueur sur les arcs et celles de la variante sur les sommets, par le biais d'une transformation de graphe. Nous proposons ensuite une formulation de programmation linéaire en nombres entiers pour aborder ce problème. Cette formulation, qui comporte un nombre exponentiel de contraintes, est une adaptation naturelle de celle conçue pour la variante du problème de bloqueur sur les arcs appliqué au flot multi-commodités discret (UMCFBP). Nous poursuivons avec une analyse polyédrale de ce modèle, offrant des perspectives sur la structure du polytope et sur ses solutions. Comme la formulation proposée n'utilise que des variables dites naturelles, elle est résolue à l'aide d'un algorithme de type Branch-and-Cut. Enfin, nous présentons plusieurs tests expérimentaux visant à évaluer la performance de cette formulation pour la variante du bloqueur sur les sommets. Nous observons, par ailleurs, que cette variante est résolue plus rapidement que la variante sur les arcs.

Conclusion

Dans cette thèse, nous nous intéressons aux problèmes de bloqueur dans les réseaux. Plus précisément, nous nous concentrons sur les problèmes de bloqueur appliqués au flot multi-commodités et à ses variantes.

Au début de la thèse, nous présentons des applications pratiques pour l'étude des problèmes de bloqueur dans les réseaux, avec un accent particulier sur les réseaux de télécommunications. Plus précisément, une application directe vise à renforcer la résilience face aux anomalies afin d'assurer la robustesse et la fiabilité des réseaux de télécommunications. Ainsi, nous concevons dans cette thèse des algorithmes adaptés à cet objectif. De plus, nous élargissons notre champ d'application pour inclure un autre cas d'usage, connu sous le nom de surveillance à l'échelle du réseau. Dans ce contexte, nous étudions le problème d'affectation de sketches et proposons des méthodes de résolution.

Dans la première partie de la thèse, nous nous penchons sur le problème de bloqueur de flot maximum (MFBP). La motivation derrière l'étude de ce

problème est multiple. Premièrement, il constitue un cas particulier du problème de bloqueur de flot multi-commodités. Deuxièmement, son importance provient de la structure spécifique du sous-problème, qui est un problème de flot maximum. Troisièmement, la vaste littérature existante sur ce sujet souligne son intérêt et son potentiel pour des recherches supplémentaires. Pour aborder ce problème, nous proposons plusieurs formulations de programmation linéaire en nombres entiers. Certaines comportent une famille exponentielle de contraintes et sont donc résolues à l'aide d'un algorithme Branch-and-Cut, adapté spécifiquement pour ce problème. Une autre formulation, avec un nombre polynomial de variables et de contraintes, a été conçue. Ces formulations sont comparées théoriquement en étudiant la force des relaxations linéaires et à travers une étude computationnelle approfondie. Aussi, nous démontrons un lien structurel entre les solutions du problème de bloqueur et celles de la variante d'interdiction du problème de flot maximum. C'est la première fois qu'une relation entre un problème de bloqueur et un problème d'interdiction appliqué à un autre problème d'optimisation est mise en lumière. Ce résultat théorique nous permet de dériver un autre algorithme pour résoudre le MFBP.

Dans la seconde partie de la thèse, nous nous penchons sur le problème de bloqueur de flot multi-commodités, en distinguant deux scénarios : l'un où le flot est continu (MCFBP), et l'autre où le flot est discret, c'est-à-dire qu'il est routé par un chemin unique pour chaque commodité (UMCFBP). Pour aborder ce challenge, nous introduisons une formulation concise qui s'inscrit dans l'espace naturel des variables de bloqueur. Cette formulation, comportant un nombre exponentiel de contraintes, peut être adaptée à plusieurs versions du problème de bloqueur de flot multi-commodités. Pour chaque variante (continue et discrète), nous étudions le problème de séparation associé et proposons plusieurs approches. Ensuite, nous présentons un algorithme Branch-and-Cut spécialement conçu pour résoudre le MCFBP et l'UMCFBP. En outre, afin de renforcer la robustesse du modèle, nous réalisons une analyse polyédrale complète. Les différentes approches et améliorations potentielles des algorithmes Branch-and-Cut sont comparées à travers une analyse expérimentale approfondie sur des ensembles d'instances synthétiques et réelles. L'objectif principal de cette étude est d'évaluer, pour chaque variante du problème, la performance de la formulation proposée et d'identifier les caractéristiques des instances qui peuvent être résolues de manière optimale.

Enfin, dans la troisième partie de la thèse, nous étendons notre recherche à la variante du problème de bloqueur sur les sommets, qui se concentre plus particulièrement sur le flot multi-commodités discret. Ce problème est désigné par V-UMCFBP. Dans cette variante, l'objectif est de supprimer des sommets du graphe au lieu de retirer des arcs. Nous démontrons, par ailleurs, l'existence d'une relation entre les deux variantes en utilisant des transformations de graphes. Plus précisément, nous démontrons que résoudre l'une des variantes équivaut à résoudre l'autre dans un graphe transformé. De plus, nous introduisons un modèle spécifique pour le V-UMCFBP, comportant un

nombre exponentiel de contraintes. Ce modèle est résolu à l'aide d'un algorithme Branch-and-Cut. Enfin, nous examinons les propriétés du polyèdre associé.

De nombreuses pistes de recherche peuvent être explorées à partir des travaux de cette thèse. Pour le problème de bloqueur de flot maximum, une étude approfondie des polyèdres des formulations proposées permettrait d'identifier les conditions dans lesquelles les inégalités définissent des facettes, ainsi que de découvrir de nouvelles inégalités valides. De plus, une exploration plus poussée de la relation entre les problèmes de bloqueur et d'interdiction pourrait fournir des informations précieuses, avec la possibilité d'étendre leur applicabilité à d'autres problèmes d'optimisation.

Pour le problème de bloqueur de flot multi-commodités, les travaux futurs pourraient se concentrer sur deux axes principaux. Le premier concerne l'aspect algorithmique, qui implique d'améliorer les heuristiques existantes ou de mettre en œuvre des stratégies de branchement plus avancées. Le second axe est plus théorique et consiste en une exploration plus approfondie des polyèdres associés aux formulations proposées pour les différentes variantes du problème. Une première piste de recherche pourrait consister à développer davantage les travaux réalisés sur la variante du bloqueur sur les sommets, en évaluant dans un premier temps l'efficacité des inégalités valides identifiées, puis en les adaptant pour la variante du bloqueur sur les arcs.

Enfin, des pistes prometteuses pour de futures recherches incluent l'intégration de contraintes du monde réel issues des réseaux de télécommunications. Plus précisément, cela pourrait impliquer de généraliser les anomalies pour tenir compte des pics de trafic, par exemple, ou d'incorporer des contraintes supplémentaires liées à la surveillance du réseau. De plus, proposer des stratégies de routage optimales qui prennent en compte les anomalies constitue une autre option à explorer. Cela pourrait notamment nécessiter de redéfinir les actions du bloqueur.

Introduction

In the fast-changing world of telecommunications, driven by increasing demand, new possibilities and challenges have appeared. One of the latest innovations is the 5G technology. The 5G connection aims to guarantee a high level of communication reliability. It can provide hundreds of billions of connections and ensure 90% more energy efficiency and 99.9% ultra-reliability (Barakovic et al. [2017]). However, among these remarkable accomplishments, the issue of signal losses in traffic transmission remains a critical concern that necessitates meticulous attention. Hence, modern telecommunication networks are extremely complex systems in which hard combinatorial optimization problems must be effectively solved in a short computing time.These problems belong to the class of *network optimization problems*. Among the most well-known problems in this class are the *maximum flow problem* and the *multi-commodity flow problem* which constitute the primary focus of this thesis.

Telecommunication networks consist of interconnected communication devices and infrastructures designed to facilitate the routing and transmission of information. These networks can be seen as graphs, where vertices represent routers connected by arcs, symbolizing transmission links. Transmission links, crucial for the routing process and facilitating the flow of data, include additional information, such as arc capacity, indicating the maximum data volume that can be transmitted through each link. Routing within these networks entails the transmission of data from a source to a destination. This process usually involves satisfying a demand, also called a commodity. In real-world scenarios, these networks exhibit extensive dimensions and they can be accompanied by a significant volume of demands. However, telecommunication networks are often confronted with anomalies and failures.

Different kinds of anomalies may arise on a telecommunication network and affect its performance. Network congestion, a common occurrence, arises when data demand exceeds network capacity, resulting in delays and reduced data transfer speeds. Equipment failures, whether due to hardware malfunctions or environmental factors, can disrupt signal transmission and compromise network reliability. Cyberattacks and security breaches introduce deliberate anomalies, compromising data confidentiality and network integrity. Signal interference from external sources, such as electromagnetic radiation or physical obstacles, can lead to signal loss. Moreover, software bugs and configuration errors can trigger unexpected behaviors and cause service interruptions. We refer the interested reader to Junior et al. [2019] for a more detailed description of network anomalies. The complexity of these anomalies highlights the need for comprehensive anomaly detection systems capable of assessing the network's ability to confront anomalies. By understanding the part of the network where anomalies and failures can arise, robust

strategies can be developed to enhance network resilience, minimize outages, and ensure seamless communication. Hence, decision-makers managing Telecommunication Networks are often faced with the question of identifying the most vital (also called most vulnerable or most critical) part of the network, which corresponds to a subset of vertices (or edges) of limited size, whose malfunctioning prevents the functionality of the network as a whole.

In this context, this work will focus on the resilience analysis of a network. More precisely, the primary goal of this study is to determine the maximum number of anomalies that may occur in the network while guaranteeing its functionality depending on a crucial property that needs to be maintained (or achieved). This property may encompass various network aspects, including constraints such as routing a specific volume of data or respecting a routing budget. This challenge belongs to a class of optimization problems called *network flow blocker problems*.

From a combinatorial optimization point of view, anomalies can be modeled with blocker notion (see Laroche et al. [2020]). Blocker notion allows to determine the minimum number of perturbations for a specific structure to be destructed. Here, the perturbations represent the anomalies, and the specific structure is a feasible solution for the network flow problem. More formally, in a network where each arc (or vertex) is associated with a removal cost, a network flow blocker problem aims to find a minimum-cost set of arcs (or vertices) to remove from the graph, rendering the network non-operational. The removed arcs (or vertices) serve as a representation of anomalies that may occur in the network and destroy transmission links (or routers). Therefore, the resolution of a network flow blocker problem provides the maximum number of anomalies the network can accommodate while still maintaining functionality. More precisely, the maximum number of anomalies is equivalent to the blocker solution value minus one.

The initial purpose of this work is to focus on multi-commodity flow problems arising in telecommunication networks with complex demand satisfaction constraints. We are interested in demands, also called commodities, having different shapes of traffic and predicted with machine learning methods. The associated network flow blocker problem is called *multi-commodity flow blocker problem* (MCFBP), where the network is represented by a multi-commodity flow problem on which a blocker problem is applied. In this context, we assume that a network is operational if there exists a multi-commodity flow in the network with a given property. The multi-commodity flow blocker problem evaluates the maximum weight of anomalies that may arise in the network so that it does not fail. Hence, a solution of the multi-commodity flow blocker problem is an indicator to define the tolerance of a network to face anomalies, which constitutes a core element for improving routing efficiency. We also study a particular case of the MCFBP, which is the *maximum flow blocker problem* (MFBP). In the MFBP, the network is represented by a maximum flow problem that aims to send a maximum amount of data from a unique source to a unique destination. To tackle these problems, we use combinatorial optimization tools, delving into various approaches including bilevel techniques that exploit the bilevel nature of network flow blocker problems, the polyhedral method initiated by Edmonds [1965] and branching algorithms.

For the maximum flow blocker problem, we present several Integer Linear Programming

(ILP) formulations designed for the first time to address this problem. The first models, featuring an exponential number of constraints, are solved through tailored Branch-and-Cut algorithms. Another ILP model with a polynomial number of variables and constraints is solved using a state-of-the-art ILP solver. The latter formulation establishes a structural connection between the maximum flow blocker problem and another related problem existing in the literature called *the maximum flow interdiction problem.* This introduces a novel approach to obtaining solutions for each problem from the other. The exact methods proposed are compared through an extensive computational campaign involving a set of synthetic and real-world instances, aiming to evaluate their performance.

We then expand our research to address the blocker variant of the multicommodity flow problem. For this purpose, we introduce an ILP formulation featuring an exponential family of constraints. This formulation is a set-covering formulation derived specifically for the multicommodity flow blocker problem and solved using a tailored branch-and-cut algorithm. Additionally, we enhance the model's robustness by developing a polyhedral approach. Several variants of the problem have been explored, including one concerning a specific instance of the multicommodity flow problem wherein the flow is considered unsplittable, meaning it is constrained to be routed through a single path for each commodity. Another variant is the vertex blocker variant of this problem, which involves removing vertices from the graph instead of arcs. All exact algorithms designed for the multicommodity flow blocker problem and its variants are computationally evaluated on a set of synthetic and real-world instances.

This thesis presents several direct applications related to Telecommunication networks. Additionally, as part of our investigation into network resilience analysis, we explored another application known as network-wide monitoring. This involves determining the optimal placement of data structures, called sketches, to monitor traffic effectively. This application can be formulated as the flow-sketch assignment problem, which is examined within this study, along with a proposed solution approach.

This dissertation is structured as follows. Chapter 1 provides notations, basic definitions, and concepts related to combinatorial optimization. Additionally, we offer a comprehensive overview of network flow problems and the blocker notion. In Chapter 2, we introduce the practical context of the problems treated in this thesis and present a solution approach for the flow-sketch assignment problem. Chapter 3 introduces formulations and algorithms to address the maximum flow blocker problem. The experimental results are detailed in Chapter 4. Chapter 5 gives a formal definition of the multicommodity flow blocker problem and presents models to solve it to proven optimality. Subsequently, in Chapter 6, we delve into the branch-and-cut algorithms developed to address the multicommodity flow blocker problem and its unsplittable variant. Chapter 7 focuses specifically on the vertex blocker variant of the multicommodity flow problem. Finally, we conclude the report with a summary of the works already done for the maximum flow blocker problem and the multi-commodity flow blocker problem. We also discuss future works, especially around the multi-commodity flow blocker problem and its applications to telecommunication networks.

Chapter 1

Preliminaries and State-of-the-art

Contents

This chapter is dedicated to the presentation of preliminary notions concerning combinatorial optimization, computational complexity, polyhedral theory, network flow problems and bilevel optimization. Precisely, we provide an overview of well known methods used in combinatorial optimization, as the Branch-and-Bound algorithm, the Branch-and-Cut algorithm, the Branch-and-Price algorithm. After giving some elements of graph theory, we give some notations related to network flow problems. These notations will be used all along the document. We then give an overview of bilevel programming with a state-of-the art on some bilevel problems such as the interdiction problem and the blocker problem.

1.1 Combinatorial optimization

Combinatorial optimisation is a type of mathematical optimization, combining applied mathematics and theoretical computer science. It is a branch of Operations research on which feasible solutions of the problem are integer solutions. In other words,

the term combinatorial optimization problem refers to the discrete structure of the feasible solutions of the problem. Due to the significant size of real-life problems, such a problem is then considered as a hard problem. Exhaustive search can not be considered and effective algorithms should be used. Algorithms and tools are developed to solve large-scale decision problems in an efficient time. These algorithms will permit to go through the decision tree efficiently by focusing on ways to exclude partial solutions in order to reach more quickly the optimal solution.
Many problems from graph theory problems are combinatorial optimization problems. For instance, let's consider the very known *Travelling Salesman Problem* (TSP). In the TSP, A salesman should visit N cities and then returns to the starting point, by reducing either total travel cost or total travel time. This problem can be modelled as a graph where vertices represent cities. If there exists a path between two cities, then it is represented by an arc on which the time and/or the cost of travel is mentioned. The TSP has been widely studied in combinatorial optimization and operations research.
Typically, the problems concerned with combinatorial optimization are those formulated as follows. Let $A = a_1, \cdots, a_n$ be a finite set called basic set where each element a_i is associated with a weight $\omega(a_i)$. Let $\mathcal{F}$ be a family of subsets of A. If $F \in \mathcal{F}$, then $\omega(F) = \sum_{a_i \in F} \omega(a_i)$ denotes the weight of F. The problem consists in identifying an element F^* of F whose weight is minimum or maximum. In other words,

$$min(\text{or max})\{\omega(F) : F \in \mathcal{F}\}. \tag{1.1}$$

Such a problem is called combinatorial optimization problem. The set F represents the set of feasible solutions of the problem.
Figure 1.1 represents the set of feasible solutions of an optimization problem containing two vector variables x and y and two family of constraints, represented by two red lines. For a combinatorial optimization problem, all feasible solutions, marked with red circles, are integer. If decision variables can take fractional values, the set of feasible solutions in represented by a green surface.

Many approaches have been developed in order to solve efficiently combinatorial optimization problems based on graph theory, linear and non linear integer programming, constraint programming or polyhedral approach. Combinatorial optimization is closely related to algorithm theory and computational complexity theory as well. In the next sections, we introduce computational issues of combinatorial optimization and present some common methods used to solve combinatorial optimization problems.

1.2 Computational complexity

Computational complexity theory, emerging from the foundations of computability theory through contributions by Cook [1971], Edmonds Edmonds [1971] and Karp Karp [1972], constitutes a segment of theoretical computer science and mathematics. Its primary objective is the classification of problems based on their level of complexity. We refer the interested reader to Garey and Johnson [1979] for a complete presentation of

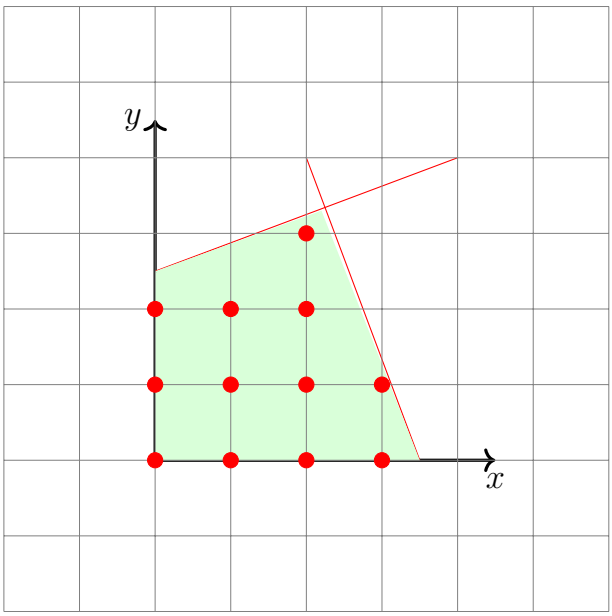


Figure 1.1: Representation of the space search of an optimization problem.

NP-completeness theory.

A *problem* entails a query characterized by input parameters, with the objective of seeking a solution. The problem is defined by its parameters and the specific criteria that solutions must fulfill. An instance of the problem is established by assigning distinct values to its input parameters.

Given an instance of a problem, an *algorithm* is a sequence of elementary operations that allows to solve the problem. We call the size of a problem the number of input parameters necessary to describe an instance. An algorithm is said to be *polynomial* if its execution time is either given by a polynomial on the size of the input, or if the number of elementary operations necessary to solve an instance of size n is bounded by a polynomial function in n. Problems that can be solved by a polynomial-time algorithm are called "easy" or "tractable" problems. We define the class P as the class gathering all the problems for which there exists some polynomial algorithm for each problem instance.

A *decision problem* is a problem with a yes or no answer. Let $\mathcal{D}$ be a decision problem and I the set of instances such that their answer is yes. $\mathcal{D}$ belongs to the class NP (Nondeterministic Polynomial) if there exists a polynomial algorithm allowing to check if the answer is yes for all the instances of I. It is clear that a problem belonging to the class P is also in the class NP. Although the difference between P and NP has not been shown, it is a highly probable conjecture. In the class NP, we distinguish some problems that may be harder to solve than others. This particular set of problems is called NP-complete. To determine whether a problem is NP-complete, we need the notion of polynomial reducibility. A decision problem $\mathcal{D}_1$ can be polynomially reduced into an other decision problem $\mathcal{D}_2$, if there exists a polynomial function f such that for every instance I of $\mathcal{D}_1$, the answer is "yes" if and only if the answer of $f(I)$ for $\mathcal{D}_2$ is "yes".

A problem $\mathcal{D}$ in NP is also NP-complete if every other problem in NP can be reduced into $\mathcal{D}$ in polynomial time. The first problem that has been shown to be NP-complete is the Satisfiability Problem (SAT) (see Cook [1971]).

With every combinatorial optimization problem is associated a decision problem. Furthermore, an optimization problem is said to be NP-Hard if its decision problem is NP-Complete. Note that most of combinatorial optimization problems are NP-Hard. One of the most efficient approaches developed to solve those problems is the so-called polyhedral approach.

1.3 Elements of polyhedral theory

The polyhedral method was initiated by Edmonds [1965] for a matching problem. It consists in describing the convex hull of problem solutions by a system of linear inequalities. The problem reduces then to the resolution of a linear program. In most of the cases, it is not straightforward to obtain a complete characterization of the convex hull of the solutions for a combinatorial optimization problem. However, having a system of linear inequalities that partially describes the solutions polyhedron may often lead to solve the problem in polynomial time. This approach has been successfully applied to several combinatorial optimization problems. In this section, we present the basic notions of polyhedral theory. The reader is referred to works of Schrijver [2003].
In the following, we recall some definitions and properties related to polyhedral theory.

A *polyhedron* P is the set of solutions of a finite linear system $Ax \leq b$, *i.e*,

$$P = \{x \in \mathbb{R}^n : Ax \leq b\} \tag{1.2}$$

Where A is a matrix $m \times n$, $b \in \mathbb{R}^m$, m and n are two positive integers.
A bounded polyhedron is called *polytope*. In other words, a polyhedron $P \subseteq \mathbb{R}^n$ is a polytope if it exists x^1, $x^2 \in \mathbb{R}^n$ such that $x^1 \leq x \leq x^2$ for all $x \in P$.

Let n be a positive integer and $x \in \mathbb{R}^n$. We say that x is a linear combination of $x_1, x_2, \cdots, x_m \in \mathbb{R}^n$ if there exists m scalar $\lambda_1, \lambda_1, \cdots, \lambda_m$ such that $x = \sum_{i \in m} \lambda_i x_i$. If $\sum_{i \in m} \lambda_i = 1$, then x is said to be a *affine combination* of $x_1, x_2, \cdots, x_m$. Moreover, if $\lambda_i \geq 0$, for all $i \in \{1, m\}$, we say that x is a convex combination of $x_1, x_2, \cdots, x_m$.
Given a set $S = \{x_1, \cdots, x_m\} \in \mathbb{R}^{m \times n}$, the *convex hull* of S is the set of points $x \in \mathbb{R}^n$ which are convex combination of $x_1, \cdots, x_m$, that is

$$conv(S) = \{x \in \mathbb{R}^n | x \text{ is a convex combination of } x_1, \cdots, x_m\} \tag{1.3}$$

Figure 1.2 represents the convex hull of a set S.

The points $x_1, \cdots, x_m \in \mathbb{R}^n$ are *linearly independents* if the unique solution of the system $\sum_{i=1}^m \lambda_i x_i = 0$ is $\lambda_i = 0$ for all $i \in \{1, m\}$. They are *affinely independents* if the unique solution solution of the system $\sum_{i=1}^m \lambda_i x_i = 0$, $\sum_{i=1}^m \lambda_i = 1$ is $\lambda_i = 0$ for all

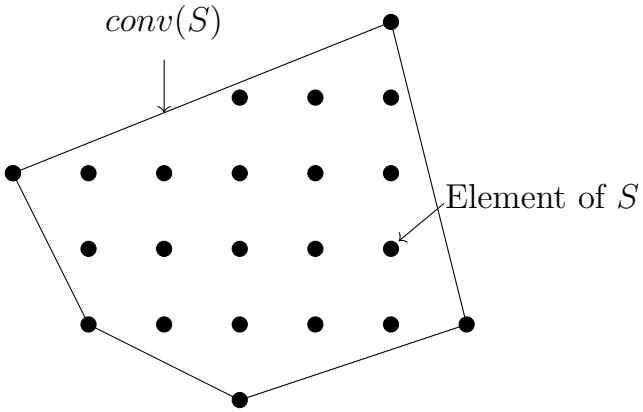


Figure 1.2: A convex hull

$i \in \{1, m\}$.

A polyhedron P is said to be of *dimension* k if it has at most $k+1$ affinely independent solutions. We note $dim(P) = k$. We also have that $dim(P) = n - rank(A^=)$, where $A^=$ is the submatrix of A of inequalities that are satisfied with equality by all the solutions of P. The polyhedron P is *full dimensional* if $dim(P) = n$.

Considering a polyhedron $P \subseteq \mathbb{R}^n$. A linear inequality $ax \leq \alpha$ is *valid* if for every solution $\overline{x} \in P$, we have $a\overline{x} \leq \alpha$. If $ax \leq \alpha$ is a valid inequality, the polyhedron

$$F = \{x \in P : ax = \alpha\} \tag{1.4}$$

is called a *face* of P. We also say that F is a face induced by $ax \leq \alpha$.
By convention, P and $\varnothing$ are faces of P. However, if $F \neq \varnothing$ or $F \neq P$, then F is a *proper face* of P.

If F is a proper face and $dim(F) = dim(P) - 1$, then F is called a *facet* of P. We also say that $ax \leq b$ induces a facet of P or is a facet defining inequality.

If P is full dimensional, then $ax \leq \alpha$ is a facet of P if and only if F is a proper face and there exists a facet of P induced by $bx \leq \beta$ and a scalar $\rho \neq 0$ such that $F \subseteq \{x \in P | bx = \beta\}$ and $b = \rho a$. In reverse, if P is not full dimensional, then $ax \leq \rho$ is a facet of P if and only if F is a proper face and there exists a facet of P induced by $bx \leq \beta$, a scalar $\rho \neq 0$ and $\lambda \in \mathbb{R}^{q \times n}$ (where q is the number of lines of matrix $A^=$) such that $F \subseteq \{x \in P | bx = \beta\}$ and $b = \rho a + \lambda A^=$.

A point x of a polyhedron P is an *extreme point* if it does not exist two solutions x^1 and x^2 of P, $x^1 \neq x^2$ such that $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$. A point $\overline{x} \in P$ is an extreme point of P if and only if $\overline{x}$ is a face of dimension 0. The polyhedron P can also be described by its extreme points.

Figure 1.3 represents a polyhedron P with extreme points, faces and facets.

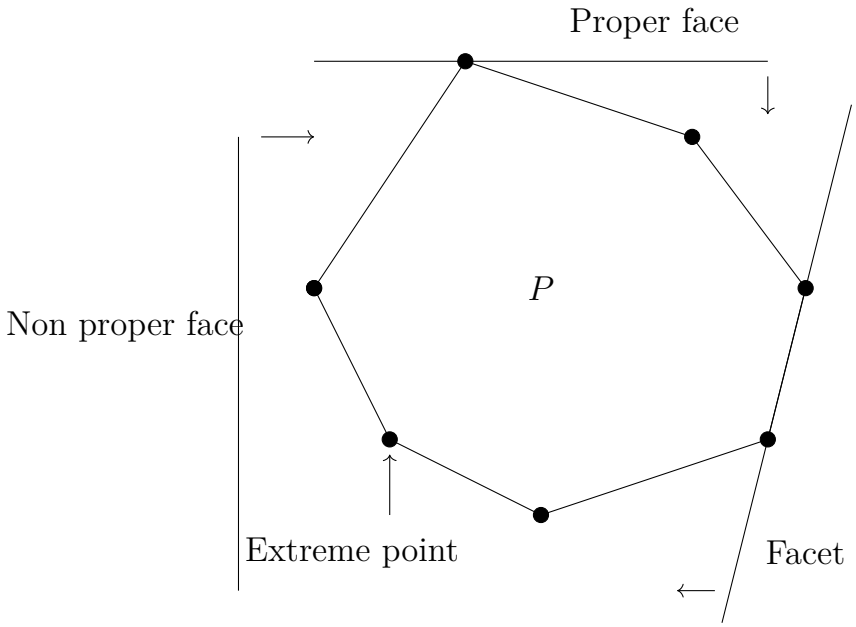


Figure 1.3: Faces, facets and extreme points

1.4 Algorithms for combinatorial optimization problems

1.4.1 Branch-and-Bound algorithm

In computer science, a **Divide-and-conquer** algorithm is a well-known algorithm that consists in dividing a problem into sub-problems, solve recursively each of them and then combine them to find a solution to the initial problem.

A **Branch-and-Bound** algorithm is based on a *Divide and Conquer* method for solving combinatorial optimization problems. The general idea is to find an optimal solution without exploring all the nodes. To prune some solutions, the algorithm uses bound on the optimal value. It falls into two parts. The first part is the branching that consists in dividing the problem into sub-problems. The second part is the bounding that consists in computing a bound of the optimal solution. The goal of this process is to ignore partial solutions.
During a Branch-and-Bound algorithm, at every node of the decision tree, the linear relaxation is solved. We recall that for an integer minimisation problem, the solution of the linear relaxation is less than or equal to the optimal integer solution. On the opposite, for an integer maximisation problem, the solution of the linear relaxation is greater than or equal to the optimal integer solution.

For a minimisation integer problem, the Branch-and-Bound method consists of several operations. At initialisation, the algorithm solve the linear relaxation of the initial problem. If an integer solution has been found, then a feasible solution has been found. Otherwise, a lower bound equal to the relaxed solution and a upper bound equal to the rounded-up integer solution, have been found. The next stage is the branching. A variable with the greatest fractional value, denoted x_i^*, is selected from the one found by the resolution of the linear relaxation. Afterwards, two new constraints for this variable are

created; $x_i \leq \lfloor x_i^* \rfloor$ or $x_i \geq \lceil x_i^* \rceil$ and two child nodes for each constraint are created. The last stage consists in solving the linear relaxation problem for each child node, considering the new constraint. At each node, a lower bound and an upper bound to the integer optimal solution are set. As before, the lower bound corresponds to the solution of the linear relaxation, and the upper bound corresponds to the existing minimum solution. The latter two steps are repeated until the optimal solution is found, i.e the smallest integer solution found during the exploration of the decision tree.

For minimization integer problems, the Branch-and-Bound method consists of several operations. At initialisation, the algorithm solves the linear relaxation of the initial problem. If an integer solution has been found, then a feasible solution is reached. In instances where an integer solution isn't achieved, a lower bound corresponding to the relaxed solution and an upper bound equal to the rounded-up integer solution are established. The subsequent step involves branching. From the variables identified through the linear relaxation resolution, the one with the most significant fractional value, denoted as x_i^*, is selected. Afterwards, two new constraints are introduced for this variable: $x_i \leq \lfloor x_i^* \rfloor$ or $x_i \geq \lceil x_i^* \rceil$, generating a pair of child nodes for each constraint. The final stage consists n solving the linear relaxation problem's solution for each child node, taking into account the recently introduced constraint. Within each node, both a lower bound and an upper bound for the integer-optimal solution are defined. The latter two steps are repeated until the optimal solution is found, i.e the smallest integer solution found during the exploration of the decision tree.

The Branch-and-Bound algorithm is not efficient when the number of constraints is exponential. Therefore, another algorithm, also based on *branching* have been developed. It's called the **Branch-and-Cut** algorithm, that we will use further to solve our problem. This algorithm is explained in the next section.

1.4.2 Cutting plane method and Branch-and-Cut algorithm

To explain **Branch-and-Cut** algorithm, the **cutting plane method** and notion of **separation** should be introduced.

Cutting plane method

Let $\mathcal{D}$ be a combinatorial optimization problem, A its basic set, ω the weight function associated with the variables of $\mathcal{D}$ and $\mathcal{S}$ the set of feasible solutions. Suppose that $\mathcal{D}$ consists in finding an element of $\mathcal{S}$ whose weight is maximum. If $F \subseteq A$, then the $0-1$ vector $x^F(a) \in \mathbb{R}^A$ such that $x^F(a) = 1$ if $a \in F$ and $x^F(a) = 0$ otherwise, is called the incidence vector of F. The polyhedron $P(\mathcal{S}) = conv\{x^S | S \in \mathcal{S}\}$ is the polyhedron of the solutions of $\mathcal{D}$. $\mathcal{D}$ is thus equivalent to the linear program $max\{cx | x \in P(\mathcal{S})\}$. It is worth noticing that the polyhedron $P(\mathcal{S})$ can be described by a set of facets defining inequalities. And when all the inequalities of this set are known, then solving $\mathcal{S}$ is equivalent to solve a linear program.

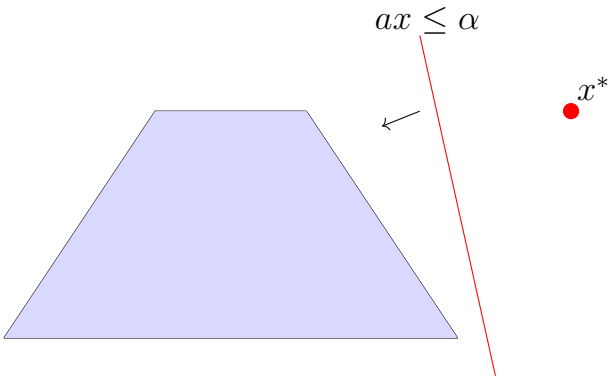


Figure 1.4: Separation process.

The polyhedral approach to addressing combinatorial optimization problems seeks to simplify the resolution of $\mathcal{D}$ by transforming it into the resolution of a linear program. This shift initiates a comprehensive investigation into the polyhedron associated with $\mathcal{D}$. Characterizing this polyhedron using a system of linear inequalities is generally complex. This challenge becomes particularly pronounced when dealing with NP-hard problems, making the prospect of achieving such a characterization quite limited. Furthermore, the number of inequalities required to define this polyhedron often becomes exponential. Consequently, even if a comprehensive description of the polyhedron can be obtained, the practical resolution remains challenging due to the large number of inequalities. Thankfully, Fortunately, a technique called the cutting plane method can be used to overcome this difficulty. This method is described in what follows.

The *cutting plane* method is based on the so-called separation problem. This consists, given a polyhedron P of $\mathbb{R}^n$ and a point $x^* \in \mathbb{R}^n$, in verifying whether if x^* belongs to P, and if this is not the case, to identify an inequality $a^T x \leq b$, valid for P and violated by x^*. In the later case, we say that the hyperplane $a^T x = b$ separates P and x^* (see Figure 1.4).

Grötschel, Lovász and Schrijver (Grötschel et al. [1981]) have established the close relationship between separation and optimization. In fact, they prove that optimizing a problem over a polyhedron P can be performed in polynomial time if and only if the separation problem associated with P can be solved in polynomial time. This equivalence has permitted an important development of the polyhedral methods in general and the cutting plane method. More precisely, the cutting plane method consists in solving successive linear programs, with possibly a large number of inequalities, by using the following steps. Let $LP = max\{cx, Ax \leq b\}$ be a linear program and LP' a linear program obtained by considering a small number of inequalities among $Ax \leq b$. Let x^* be the optimal solution of the latter system. We solve the separation problem associated with $Ax \leq b$ and x^*. This phase is called the separation phase. If every inequality of $Ax \leq b$ is satisfied by x^*, then x^* is also optimal for LP. If not, let $ax \leq \alpha$ be an inequality violated by x^*. Then we add $ax \leq \alpha$ to LP' and repeat this process until an optimal solution is found. The following algorithm summarizes the different cutting plane steps.

It is worth noticing that a cutting-plane algorithm might not always achieve an optimal

Algorithm 1 A cutting plane algorithm

Input: A linear program LP and its system of inequalities $Ax \leq b$
Output: Optimal solution x^* of LP
Consider a linear program LP' with a small number of inequalities of LP;
Solve LP' and let x^* be an optimal solution;
Solve the separation problem associated with $Ax \leq b$ and x^*;
if an inequality $ax \leq \alpha$ of LP is violated by x^* **then**
    Add $ax \leq \alpha$ to LP';
    Repeat step 2;
else
    x^* is the optimal solution for LP;
    return x^*
end if

solution for the core combinatorial optimization problem. In this case, a Branch-and-Bound algorithm can be used to tackle the issue. This leads to a Branch-and-Cut algorithm, in which a series of separation problems are solved for each node of the decision tree.

Branch-and-Cut algorithm

Consider again a combinatorial optimization problem $\mathcal{D}$ and suppose that $\mathcal{D}$ is equivalent to:

$$max\{cx|Ax \leq b, x \in \{0,1\}^n\}$$

Where $Ax \leq b$ has a large number of inequalities. A Branch-and-Cut algorithm starts by creating a Branch-and-Bound tree whose root node corresponds to a linear program $LP_0 = max\{cx|A_0x \leq b_0, x \in \mathbb{R}^n\}$, where $A_0x \leq b_0$ is a subsystem of $Ax \leq b$ having a small number of inequalities. Then, we solve the linear relaxation of $\mathcal{P}$ that is $LP = \{cx|Ax \leq b, x \in \mathbb{R}^n\}$ using a cutting plane algorithm whose starting from LP_0. Let x_0^* denote its optimal solution and $A_0'x \leq b_0'$ the set of inequalities added to LP_0 at the end of the cutting plane phase. If x_0^* is integral, then it is optimal. If x_0^* is fractional, then we perform a branching phase. This step consists in choosing a variable, say x^1, with a fractional value and adding two nodes D_1 and D_2 in the Branch-and-Cut tree. The node P_1 corresponds to the linear program $LP_1 = max\{cx|A_0x \leq b_0, A_0'x \leq b_0', x^1 = 0, x \in \mathbb{R}^n\}$ and $LP_1 = max\{cx|A_0x \leq b_0, A_0'x \leq b_0', x^1 = 1, x \in \mathbb{R}^n\}$. We then solve the linear program $\overline{LP_1} = max\{cx|Ax \leq b, x^1 = 0, x \in \mathbb{R}^n\}$ (resp. $\overline{LP_2} = max\{cx|Ax \leq b, x^1 = 1, x \in \mathbb{R}^n\}$) by a cutting plane method, starting from LP_1 (resp. LP_2). If the optimal solution of $\overline{LP_1}$ (resp. $\overline{LP_2}$) is integral then, it is feasible for P. Its value is then a lower bound of the optimal solution of $\mathcal{D}$, and the node D_1 (resp. D_2) becomes a leaf of the Branch-and-Cut tree. If the solution is fractional, then we select a variable with a fractional value and add two children to the node D_1 (resp. D_2), and so on.

It is worth noticing that sequentially adding constraints of type $x_i = 0$ and $x_i = 1$ where x_i is a fractional variable, may lead to an infeasible linear program at a given node of the Branch-and-Cut tree. Or, if it is feasible, its optimal solution may be worse than

the best known lower bound of the problem. In both cases, that node is pruned from the Branch-and-Cut tree. The algorithm ends when all nodes have been explored and the optimal solution of $\mathcal{D}$ is the best feasible solution given by the Branch-and-Bound tree.

This algorithm can be improved by computing a good lower bound of the optimal solution of the problem. This lower bound, once determined, serves as a criterion for the algorithm to discard nodes that cannot lead to an enhancement of this established lower bound. This would permit to reduce the number of nodes generated in the Branch-and-Cut tree, and hence, reduce the time used by the algorithm. Moreover, this lower bound can be improved through a process involving the comparison of a feasible solution at each node within the Branch-and-Cut tree, particularly when the solution derived at the root node is fractional. This technique is commonly known as a primal heuristic. Its purpose is to generate a viable solution for $\mathcal{D}$ based on the solution acquired at a specific node within the Branch-and-Cut tree. This approach is especially useful when the solution at this node is fractional, and hence infeasible for $\mathcal{D}$. Moreover, the weight of this solution must be as best as possible. When the computed solution surpasses the current best-known lower bound, it may significantly reduce the number of generated nodes, as well as the CPU time. This also guarantees to have an approximation for the optimal solution of $\mathcal{D}$ before visiting all the nodes of Branch-and-Cut tree, for example when a CPU time limit has been reached.

The Branch-and-Cut approach has proven highly effective in addressing a range of combinatorial optimization problems that are typically regarded as challenging. It is worth noticing that a strong understanding of the problem's associated polyhedron, together with efficient separation algorithms (both exact and heuristic), can significantly enhance the efficiency of this approach. Additionally, the cutting plane method is efficient when the number of variables remains polynomial. However, when the number of variables is exponential, alternative strategies like column generation are more likely to be used. An overview of this method is provided in the next section.

1.4.3 Column generation and Branch-and-Price

Column generation

Compact formulations of combinatorial optimization problems often provide a weak linear relaxation. Those problems require then further formulations, whose linear relaxation is closer to the convex hull of feasible solutions. Those reformulations may have a huge number of variables, so that one can not consider them explicitly in the model. we describe a method that suits well to such reformulation, that is the so-called column generation method.

Combinatorial optimization problems often give rise to compact formulations characterized by a weak linear relaxation. As a consequence, these problems require further formulations, whose linear relaxation is closer to the convex hull of feasible solutions. These reformulations can potentially introduce a huge number of variables, that can not be explicitly in the model. The column generation method proves to be particularly well-suited for addressing such types of reformulations.

The column generation method is employed to tackle linear programs featuring an extensive set of variables, focusing only on a subset of these variables. This innovative technique was first introduced by Dantzig and Wolfe in 1960 (Dantzig and Wolfe [1960]), aimed at addressing challenges arising in order to solve problems that could not be handled efficiently because of their size, encompassing considerations of CPU time and memory consumption.

Column generation is generally used either for problems whose structure is suitable for a Dantzig-Wolfe decomposition, or for problems with a large number of variables. The overall idea of column generation is to solve a sequence of linear programs with a restricted number of variables (also referred to as columns). The algorithm starts by solving a linear program having a small number of variables, and such that a feasible solution for the original problem may be identified using this basis. At each iteration of the algorithm, we solve the so-called pricing problem whose objective is to identify the variables which must enter the current basis. These variables are characterized by a negative reduced cost. The reduced cost associated with a variable is computed using the dual variables associated with the constraints of the problem. We then solve the linear program obtained by adding the generated variables, and repeat the procedure until no variable with reduced cost can be identified by the pricing problem. In this case, the solution obtained from the last restricted program is optimal for the original model. The main step of column generation procedure is summarized in Algorithm 2.

Algorithm 2 A column generation algorithm

Input: A linear program MP (Master Problem) with a huge number of variables
Output: Optimal solution x^* of MP
Consider a linear program RMP (Restricted Master Problem) including only a small subset of variables of the MP;

1. Solve RMP and let x^* be an optimal solution
2. Solve the pricing problem associated with the dual variables obtained by the resolution of the RMP

if there exists a variable x with a negative reduced cost **then**
    add x to RMP;
    go to step 2;
else
    x^* is the optimal solution for MP;
    return x^*
end if

The column generation method can be seen as the dual of the cutting plane method since it adds columns (variables) instead of rows (inequalities) in the linear program. Furthermore, the pricing problem may be NP-hard. One can then use heuristic procedures to solve it.

Branch-and-Price algorithm

The solution obtained by a column generation procedure may not be integer. Therefore, to solve an integer programming problem, the column generation method has to be integrated within a Branch-and-Bound framework. This is known a Branch-and-Price algorithm. Branch-and-Price is similar to Branch-and-Cut approach, except that procedure focuses on column generation rather than row generation. In fact, generating variables (pricing) and adding inequalities (cutting plane) are complementary operations to strengthen the linear relaxation of a integer programming formulation.

The Branch-and-Price procedure works as follows. Each node of the Branch-and-Bound tree is solved by column generation, so that variables may be added to improve the linear relaxation of the current LP. The branching phase occurs when no columns price out to enter the basis and the solution of the linear program is not integer.

Note that, at each node of the Branch-and-Price tree, column generation may be combined with cutting plane approach, to tighten the LP relaxation of the problem. In this case, the algorithm is called Branch-and-Cut-and-Price algorithm. Such a method can be difficult to handle, since adding valid inequalities to the initial model may change the structure and complexity of the pricing problem.

1.5 Network flow problems

In this section, we present fundamental definitions and notations from graph theory that are specifically utilized in the formulation of network flow problems.

1.5.1 Graph theory

A **graph** is denoted $G = (V, E)$, where V is the set of vertices and E is the set of edges. If e is an edge with a direction, *i.e*, e connects a vertex u to a vertex v, then the edge is called an *arc* written $a = (u, v)$ and the graph is said to be *directed*. In the remainder of this document, we consider only directed graphs.
A directed graph is denoted $G = (V, A)$ with $m = |A|$ arcs and $n = |V|$ vertices. Each arc $a \in A$ is associated with a positive integer capacity denoted $c_a \in \mathbb{Z}_+$. We say that $a = (u, v)$ is an outgoing arc of u and an incoming arc of v. Two vertices are called *neighbors* if they are connected by an arc. Given a vertex $v \in V$, $N(v)$ designates the neighbors of v, $\delta^-(v)$ designates the incoming arcs of v and $\delta^+(v)$ designates the outgoing arcs of v, *i.e*,

$$\delta^-(v) = \{a \in A | a = (u, v), \forall u \in V\}$$

$$\delta^+(v) = \{a \in A | a = (v, u), \forall u \in V\}$$

We define a *path* p as an ordered set of $|p|$ distinct vertices $\{v_1, \cdots, v_{|p|}\}$ such that for all $i \in \{1, \cdots, |p| - 1\}$, (v_i, v_{i+1}) is an arc. We denote by $A(P)$ the set of arcs

$(v_i, v_{i+1}), i \in \{1, \cdots, |p|-1\}$. An $s-t$ path is a path P that connects a vertex $s \in V$, called *source*, to a vertex $t \in V$, called *destination*. In other words, an s-t path, often referred to as just a path, is a sequence of vertices such that $v_1 = s$ and $v_{|p|-1} = t$.

1.5.2 Flow notations

Given a directed graph $G = (V, A)$ with a source $s \in V$ and a destination $t \in V$, the flow from s and t, also called $s-t$ flow, is defined as a function $f : A \to \mathbb{R}^+$ that assigns to each arc $a \in A$ a positive integer value such that $f(a) \leq c_a$ and for every vertex u that is not the source or the destination, $\sum_{a \in \delta^+(u)} f(a) = \sum_{a \in \delta^-(u)} f(a)$. The value of a flow f, denoted $v(f)$, is equal to the sum of all values affected to the arcs outgoing s, which is equal to the sum of all values affected to the arcs incoming t, *i.e*, $v(f) = \sum_{a \in \delta^+(s)} f(a) = \sum_{a \in \delta^-(t)} f(a)$.
A *feasible flow* from the source s to the destination t, or simply a flow, has two respect two typologies of constraints, the *capacity constraints* and the *flow conservation constraints*. The capacity constraints impose that the value of the flow passing through an arc a should be less than or equal to the maximum capacity c_a of the arc, i.e.,

$$f(a) \leq c_a, \quad \forall a \in A.$$

The flow conservation constraints impose that the total value of the flow entering a vertex is equal to the total value of the flow outgoing this vertex, i.e.,

$$\sum_{a \in \delta^+(u)} f(a) = \sum_{a \in \delta^-(u)} f(a), \quad \forall u \in V.$$

Without loss of generality, we consider graphs in which the source has only one entering arc from the destination, i.e., $\delta^-(s) = \{(t, s)\}$ and the destination has only one outgoing arc, i.e., $\delta^+(t) = \{(t, s)\}$.

A *network* is formally defined as a directed graph denoted by $G = (V, A)$. It may contain a distinct source node, denoted as $s \in V$, as well as a singular destination node, denoted as $t \in V$. Furthermore, the network has the capability to accommodate multiple sets of source and destination pairs, which are alternatively referred to as *commodities*. A commodity, denoted k, is a quadruplet (s_k, t_k, b_k, r_k), where $s_k \in V$ is the source, $t_k \in V$ is the destination, $b_k \in \mathbb{R}^+_*$ is the bandwidth and $r_k \in \mathbb{R}^+_*$ is a reward obtained when sending one unit of flow from s_k to t_k. A commodity k is said to be *fully satisfied* if there exists a flow f_k from s_k to t_k such that $v(f) = d_k$. We denote by $\mathcal{K}$ the set of commodities. On the other hand, a commodity is said to be *partially satisfied* if $v(f)$ is striclty greater than 0 and strictly smaller than d_k. Finally, if $v(f)$ is equal to 0, then the commodity is said to be *unsatisfied*.

1.5.3 State-of-the-art on network flow problems

Network flow problems are a class of optimization problems that aims to find feasible flows in a network, according to a given objective. One of the most well known network flow problem is the *Maximum flow problem*, presented below.

The *Maximum Flow Problem*, or *Maximum $s-t$ flow Problem*, is one of the most studied optimization problem, behind intense research since its introduction in the fiftieths. We refer the interested reader to Schrijver [2002] for an historic overview on the MFP. The problem has countless applications in several domains and it has been originally proposed to compute the maximum flow, from a given city to another, on a rail network.
The MFP finds important applications also in telecommunication networks where an optimal MFP solution allows to determine the maximum traffic (flow) that can be sent between a given pair of origin destination router (a specific pair of vertices). This problem is particularly important in the planning/strategical phase when, for a given telecommunication network, the maximum amount of data that can be accepted needs to be evaluated, see Ahuja et al. [1993].

Formally, given a directed graph $G = (V, A)$ containing a source s and a destination t (i.e., $s, t \in V$), the MFP calls for determining the maximum s-t flow, or simply flow, in G from the source s to the destination t that respects the arc capacities. The MFP can be solved in polynomial time and several efficient exact algorithms are proposed in the literature, see e.g., Ahuja et al. [1993]. The first pseudo-polynomial labelling algorithm is described in Ford and Fulkerson [1956]. It runs in $O(n\,m\,C)$ where C is the largest arc capacity. To avoid the dependency on C, two polynomial time algorithms have been proposed in Edmonds and Karp [2003]. The first one, called the capacity scaling algorithm runs in $O(n\,m\,\log C)$ and the second, called the successive shortest path algorithm, runs in $O(n^2\,m)$. The most efficient algorithm in practice, especially for dense and large graphs, is the highest-label preflow-push algorithm developed by Goldberg and Tarjan [1988] which runs in $O(n^2\,m^{1/2})$ time.

Another way to solve the MFP is to use a *Linear Programming* (LP) formulation where a continuous non-negative variable $y_a \geq 0$, defined for each arch $a \in A$, represents the flow $f(a)$ on the arc a. Therefore, the Maximum Flow Problem is equivalent to the following LP model:

$$\zeta(\text{MFP}) \;=\; \max \quad \sum_{a \in \delta^+(s)} y_a \tag{1.5a}$$

$$\sum_{a \in \delta^+(u)} y_a - \sum_{a \in \delta^-(u)} y_a = 0 \qquad u \in V, \tag{1.5b}$$

$$y_a \leq c_a \qquad a \in A, \tag{1.5c}$$

$$y_a \geq 0 \qquad a \in A, \tag{1.5d}$$

The objective function (1.5a) represents the flow from the source s to the destination t. Constraints (1.5b) are the *flow conservation* constraints, imposing that for each vertex $u \in V$ the entering flow is equal to the leaving one. Constraints (1.5c) impose the maximum capacity for the flow on each arc $a \in A$. Finally, constraints (1.5d) express the definition set of the arc-flow variables $\mathbf{y}$.

There exists an equivalence between the MFP and another very known optimization problem, called the Minimum Cut Problem, see Ahuja et al. [1993]. This relation has

been largely reused since its discovery. First, in order to introduce the Minimum Cut Problem (MCP), we denote by $\delta(U)$ a *cut* of a graph $G = (V, A)$, which corresponds to the set of arcs of G with the tail in U and the head in $V \backslash U$. In this way, the *Minimum Cut Problem* consists in finding a $s - t$ cut $\delta(U)$ such that *i*) the source s belongs to U; *ii*) the destination t belongs to $V \backslash U$; *iii*) the sum of the capacities of the arcs $(u, v) \in \delta(U)$ is minimum. For sake of brevity, we write cut for $s - t$ cut. Second, due to the equivalence between the MFP and the MCP, the MCP can also be solved in polynomial time. Third, the dual of the MFP formulation (1.5) provides a valid LP formulation for the MCP and the dual variables of the constraints of (1.5) take integer values in any of its optimal solution. This is due to the fact that the system of constraints (1.5b) and (1.5c) is Totally Unimodular (TU) see Schrijver [2003]. Precisely, the dual of model (1.5) leads to find a cut $\delta(U)$. Let $\alpha_u \in \mathbb{R}$ $(u \in V)$ be the dual variable associated with constraint (1.5b). If the variable α_u is equal to 1, then it indicates that vertex u is in the subset U (otherwise vertex u belongs to $V \backslash U$). Let $\beta_a \geq 0$ $(a \in A)$ be the non-negative dual variable associated with constraint (1.5c). If $\beta_a > 0$, then arc a belongs to the cut $\delta(U)$. The LP formulation of the MCP reads as follows:

$$\zeta(\text{MCP}) = \min \quad \sum_{a \in A} c_a \, \beta_a \tag{1.6a}$$

$$\beta_{uv} + \alpha_v - \alpha_u \geq 0 \qquad (u, v) \in A, \tag{1.6b}$$

$$\alpha_s - \alpha_t \geq 1 \tag{1.6c}$$

$$\beta_a \geq 0 \qquad a \in A. \tag{1.6d}$$

We denote by $\zeta(\text{MCP})$ the optimal solution value of formulation (1.6) and we recall that variables α and β can also be casted as binary variables (as discussed above). The objective function (1.6a) minimizes the total capacity of the arcs in the cut. Constraints (1.6b) force an arc (u, v) to be in the cut, i.e., $\beta_{uv} = 1$, if $\alpha_u = 1$ and $\alpha_v = 0$. Constraint (1.6c) together with the objective function imposes that $\alpha_t = 0$ and $\alpha_s = 1$, i.e., the source belongs to the subset U and the destination belongs to the subset $V \backslash U$. Finally, constraints (1.6d) express the nature of the dual variables β.

A fundamental min-max relation links the MFP to the MCP, see e.g., Ahuja et al. [1993]. Precisely, the *max-flow min-cut theorem* states that the optimal value of the MFP and the MCP coincides, i.e., $\zeta(\text{MCP}) = \zeta(\text{MFP})$. In addition, by duality, any feasible MFP solution provides a valid lower bound for the MCP, and, vice versa, any feasible MCP solution provides a valid upper bound for the MFP.

Figure 1.5 represents a graph with 9 vertices and 17 arcs. For each arc $a \in A$, the figure reports two values separated by "/". The first one in blue (to the left of "/") corresponds to the flow on the arc of an optimal MFP solution. The second one in black (to the right of "/") is the capacity c_a of the arc. The minimum cut $\delta(U)$ is represented by thicker red lines and the grey shaded surface surrounds, on both sides of the minimum cut, the set of vertices U and $V \setminus U$. In this example, we have: $\zeta(MFP) = \zeta(MCP) = 36$, $\delta(U)$ is given by the set of arcs (v_1, v_5), (v_4, v_6), (v_4, v_7) and (v_3, v_7), U is the set of vertices $\{s, v_1, v_2, v_3, v_4\}$ and $V \backslash U = \{v_5, v_6, v_7, v_8\}$.

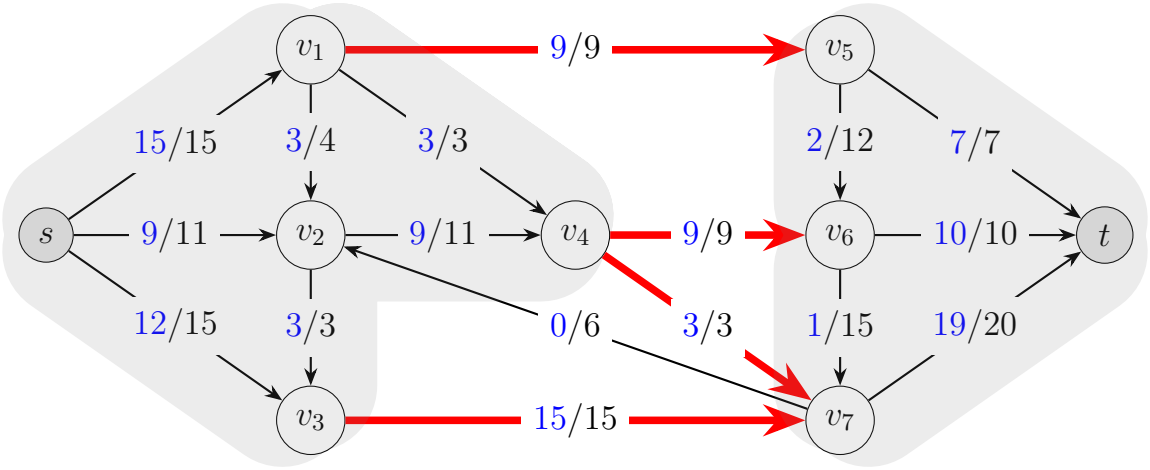


Figure 1.5: An example of Maximum flow and Minimum Cut.

Another very known network flow problem is the *minimum cost flow problem*, which is a generalization of the maximum flow problem. In the minimum cost flow problem, we are given a directed graph $G = (V, A)$, where each arc $a \in A$ has a positive integer capacity c_a and a positive integer flow cost ω_a, and a flow value denoted by F. As previously, a minimum cost $s - t$ flow problem assigns to each arc $a \in A$ a flow value, respecting the flow conservation constraints and the capacity constraints. The cost of a flow f, denoted by $p(f)$, is then defined by summing the cost per unit of flow for each arc in the network, i.e,

$$p(f) = \sum_{a \in A} \omega_a f(a).$$

The minimum cost $s-t$ flow problem, or simply the minimum cost flow problem, aims to find a flow of value F between the source s and the destination t such that the total cost $p(f)$ is minimum. Using the continuous non-negative variables y_a introduced previously, for each arc $a \in A$, the minimum cost flow problem can be modeled by the following LP formulation:

$$\min \left\{ \sum_{a \in A} y_a \, \omega_a : (1.5b), (1.5c), \sum_{a \in \delta^+(s)} y_a = F, y_a \geq 0, \ \forall \, a \in A \right\}. \tag{1.7}$$

The model presents the same constraints as Model (1.5), i.e., the flow conservation constraints and the capacity constraints, with an additional constraint $\sum_{a \in \delta^+(s)} y_a = F$ imposing the value of the flow. The objective function minimizes the total flow cost.

Let us go back to the example graph used in Figure 1.5 to represent a maximum flow and a minimum cut. The next figure represents the same graph and it is used to show the solution of a minimum cost flow problem. As previously, for each arc $a \in A$, the figure reports two values separated by "/". The first one in blue corresponds to the flow on the arc of an optimal solution for the minimum cost flow problem and the second one in black is the capacity c_a of the arc. A third value is added, separated from the others by a semicolon ";" and representing the cost per unit of flow passing though the arc. The flow value is set to $F = 25$. Figure 1.6 represents a flow of value 25 with a minimum total cost, equal to 133.

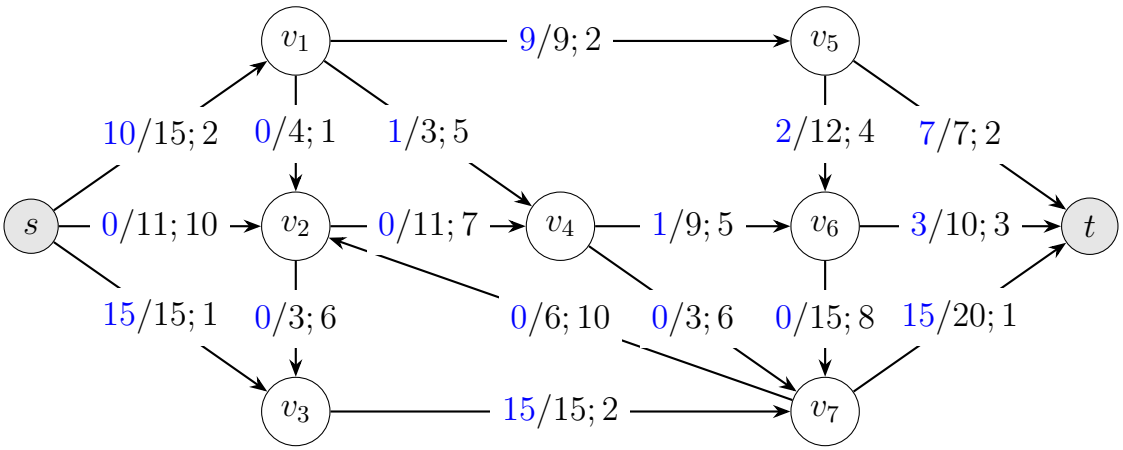


Figure 1.6: An example of Minimum Cost Flow.

The generalization of network flow problems involves the incorporation of multiple sources and destinations, giving rise to what is known as *multicommodity flow problems*.

Multicommodity flow problems

Multi-commodity flow problems (MCFPs) are widely studied network flow problems arising in telecommunication networks. Given a set of commodities, a MCFP aims to find a set of flows to satisfy a collection of commodities, respecting a specified objective function. A multicommodity flow problem (MCFP) is said to be *unsplittable* if, for each commodity, the flow is routed through a unique path connecting the source to the destination. In this case it is denoted by *uMCFP*. Otherwise, if the flow can be split into several paths, the MCFP is said to be *splittable*.

In literature, many articles deal with the multi-commodity flow problem, and how to solve it in an efficient way. Indeed, MCFPs have been the object of interest since the 1960s, when they were introduced in Ford and Fulkerson [1962] and Hu [1963]. Their applications are diverse, encompassing transportation, logistics and telecommunication networks (see Zhang et al. [2023]). There are two broad categories of multi-commodity flow problems. The first category (*min-MCFP*) aims to minimize the total cost while routing b_k units of flow for each commodity $k \in K$ from the source s_k to the destination t_k. The second category (*max-MCFP*) aims to maximize the total flow between all pairs of sources and destinations. The MCFP can generally be solved in polynomial time, see Tardos [1986]. However, it can be NP-Hard under some routing properties. This is the case of the uMCFP, which is NP-Complete in its decision version and accordingly, $\mathcal{NP}$-hard in its optimization version, see e.g., Kleinberg [1996] and Dinitz et al. [1998].

In 1978, two surveys were published on multi-commodity flow problems, see e.g., Assad [1978] and Kennington [1978]. In Assad [1978], several methods have been proposed to solve MCFPs with linear and non-linear routing costs. For a min-MCFP with linear costs, the author used primal-dual methods, which involve constructing a feasible dual solution and iteratively improving it until a primal solution is found. Another method has been presented to solve a max-MCFP. The latter is based on a formulation of the

MCFP where each commodity is associated with a set of paths that can be used to route the commodity. This formulation has an exponential number of variables and it is solved using column generation and decomposition algorithms. As explained previously, in a column generation algorithm, only a subset of the variables, i.e., paths, are initially considered, and additional variables are generated iteratively as needed to improve the solution until the optimal solution is found. Decomposition algorithms decompose the problem into smaller subproblems that can be solved more efficiently. In Kennington [1978], the author mostly focuses on subgradient methods and partitioning techniques to solve the MCFP with linear costs. Another survey is dedicated only to non-linear convex multi-commodity flow problems, see Ouorou et al. [2000], for further details. The algorithms mentioned in this survey include flow deviation methods, projection methods, cutting plane algorithms and the proximal decomposition method. More recently, the survey conducted by Wang [2018a] presents the min-MCFP along with a review of its applications. The follow-up extension of this survey, see Wang [2018b], covers various methods to solve the MCFP with linear costs. In addition to the conventional methods mentioned previously, this paper presents some new techniques to solve the MCFP and the uMCFP, including heuristics, approximation algorithms or interior-point methods. These methods can be used to find good quality solutions to the MCFP, especially for large-scale problems. Lately, a constraint-programming based branch-and-price-and-cut framework has been developed in Zhang et al. [2024] to solve bipath multicommodity flow problems. On the other hand, many works in the literature have also been dedicated to the uMCFP. In Barnhart et al. [2000], the authors propose a *Branch-and-Cut-and-Price* approach to solve the uMCFP. This algorithm was further improved in Alvelos and Carvalho [2003], where a Dantzig-Wolfe decomposition is used to solve the LP relaxations more efficiently. The Dantzig-Wolfe decomposition involves decomposing the original LP into smaller subproblems, which can be solved in parallel using a master problem and a set of pricing problems. In addition, numerous approximation algorithms have been investigated in the literature. For a comprehensive survey of approximation algorithms for the uMCFP, we refer the interested reader to Kolliopoulos and Stein [2001]. Lately, the unsplittable multicommodity flow problem have been solved via quantum computing in Martin and Martin [2023].

1.6 State-of-the-Art on bilevel problems

This section focuses on bilevel programming, specifically exploring two classes of problems, namely the interdiction problems and the blocker problems.

1.6.1 Bilevel programming

A **bi-level optimization** problem is an optimization problem in which one of the constraint is also an optimization problem. We refer the interested reader to Kleinert et al. [2021] and Beck et al. [2023] for surveys on bilevel optimization.

A bi-level problem contains two problems; the *upper-level optimization problem* also called *master problem* and the *lower-level optimization problem*, also called *follower*

problem. The global mathematical formulation of a bi-level problem is as follows :

$$\begin{aligned} &\min_{s.t} F(x,y) \\ &G_i(x,y) \leq 0, i = \{1, \cdots, I\} \\ &y \in argmin\{f(x,y) | g_j(x,y) \leq 0, j = \{1, \cdots, J\}\} \end{aligned} \tag{1.8}$$

where $F(x,y)$ and $f(x,t)$ are the upper-level objective function (leader) and lower-level objective function (follower), respectively. $G_i(x,y)$ and $g_j(x,y)$ are the inequality constraints of the upper and lower level problems, respectively. Finally x and y are respectively the master and the follower decision variable vectors.

Bilevel optimization has received a lot of attention recently, due to its intricate nature and wide-ranging applications. Indeed, bilevel optmization can appear in many practical problems in several fields such as transportation, economics, defense, and many others. It is also often used in network design problems.

A notable example of bilevel optimization can be found in the context of Stackelberg games, which are used to model hierarchical decision-making scenarios. In a Stackelberg game, one player, known as the leader, makes decisions first, taking into account the actions of another player, the follower. The leader aims to optimize its objective function while considering the follower's response. On the other hand, the follower seeks to optimize its objective function, based on the leader's decisions. The Stackelberg game is a concept of game theory where the two players interact strategically. In literature, many techniques have been developed to solve complex Stackelberg games efficiently. We refer the interested reader to Stackelberg [1952] for further explanations regarding this concept and the solution approaches.

In the following sections, we focus on two specific class of bilevel optimization problems, the interdiction problems and the blocker problems.

1.6.2 Interdiction problems

An interdiction problem is a bi-level optimization problem that consists in performing a set of operations in the underlying problem to destruct its structure. The cost of the operations is bounded by a budget. As an illustrative example, we examine a class of interdiction problems known as *network interdiction problems*, which involves applying an interdiction problem to a network flow problem. The master variable vector, referred to as the interdictor variable vector, aims to minimize the network's profit while respecting the constraint imposed by the interdiction budget.

Interdiction problems have been widely studied in literature. Some relevant applications of interdiction problems include the firefighting problem (García-Martínez et al. [2015]), the control of infections in hospitals (Assimakopoulos [1987]), the allocation of protective resources (Cappanera and Scaparra [2011]), the protection and analysis of supply chain disruptions (Snyder et al. [2016]), or the detection of drug smuggling (Wood [2011]). As a consequence, interdiction problems have been applied to several well known optimization problems, such as the maximum flow problem (see Wood [1993]), the maximum clique problem (see Furini et al. [2019]), the shortest path problem (see Israeli and Wood

[2002a]). In what follows, we describe more formally some interdiction problems that have been studied in the literature. Specifically, we present the works already done on the *maximum flow interdiction problem*, the *multi-commodity flow interdiction problem* and the *maximum clique interdiction problem*.

The maximum flow interdiction problem

In this section, we introduce the *maximum flow interdiction problem* (MFIP), presenting its principal structural properties and mathematical programming formulations from the literature.

The MFIP can be formally described as follows. Given a directed graph $G = (V, A)$ containing a source $s \in V$ and a destination $t \in V$, a set of capacities $c_a \in \mathbb{Z}_+$, a set of interdiction costs $q_a \in \mathbb{Z}_+$ associated to each arc $a \in A$ and an interdiction budget Ψ, the problem consists in finding a subset of arcs of a cost no larger than Ψ and such that the maximum flow between the source and the destination in the interdicted graph is minimum.

Let $\mathbf{w}$ be a vector of binary variables associated with the set of arcs A of a graph G, each variable encoding whether the corresponding arc is interdicted or not. The entire set of feasible *interdiction policies* reads as follows

$$\mathcal{W} = \left\{ \mathbf{w} \in \{0,1\}^m : \sum_{a \in A} q_a \, w_a \leq \Psi \right\}. \tag{1.9}$$

A binary realization $\mathbf{w} \in \{0,1\}^m$ of the first-level variables, is called an *interdiction policy* and it generates a *non-interdicted graph* $\mathcal{G}_{NI}(\mathbf{w}) = \big(V, \mathcal{A}_{NI}(\mathbf{w})\big)$, i.e., the graph induced by the set of non-interdicted arcs $a \in A$ with $w_a = 0$ (denoted $\mathcal{A}_{NI}(\mathbf{w})$).

In Wood [1993], the author presents a bilevel model to solve the MFIP and proposes a procedure to obtain an ILP formulation from the bilevel model.

The MFIP can be stated as the following bilevel optimization model:

$$\zeta(\text{MFIP}) \;=\; \min_{\mathbf{w} \in \{0,1\}^m} \quad \vartheta(w) \tag{1.10a}$$

$$\sum_{a \in A} q_a \, w_a \leq \Psi \tag{1.10b}$$

$$\text{where} \quad \vartheta(w) = \max_{\mathbf{y} \,\in\, \mathbb{Q}_+^m} \quad \sum_{a \,\in\, \delta^+(s)} y_a \tag{1.10c}$$

$$\sum_{a \,\in\, \delta^+(u)} y_a - \sum_{a \,\in\, \delta^-(u)} y_a = 0, \qquad \forall\, u \in V, \tag{1.10d}$$

$$y_a \leq c_a \, (1 - w_a), \qquad \forall\, a \in A. \tag{1.10e}$$

$$y_a \geq 0, \qquad \forall\, a \in A. \tag{1.10f}$$

In this model, the leader (or first level) problem is given by the first $0-1$ program stated by Equation ((1.10a)-(1.10b)) and the follower (or second level) problem is given by the inner maximization program stated by ((1.10c)-(1.10f)). The leader permits to determine the set of interdicted arcs, while the follower aims at computing the maximum flow in $\mathcal{G}_{NI}(\mathbf{w})$. The variable $\vartheta(w)$ represents the value of the maximum flow in the non-interdicted graph $\mathcal{G}_{NI}(\mathbf{w})$. This value is obtained as the optimal value of the *follower problem*. Constraint (1.10b) guarantees that the interdiction budget is respected. Constraints (1.10d) are the flow conservation constraints (as in (1.5)) and constraints (1.10e) prevent the flow to be routed on interdicted arcs.

In order to obtain a compact ILP single level model, a well-known technique, used when the follower problem can be modelled as a linear program, consists in dualizing it. To this end, we have first to remove the dependency of the $\mathbf{w}$ variables in constraints (1.10e). In other words; the link between the leader and the follower problems must be only expressed in the objective function. More precisely, given an interdiction policy $\mathbf{w} \in \mathcal{W}$, Theorem 2 of Cormican et al. [1998] shows that the follower problem can be restated as follows:

$$\vartheta(w) \;=\; \max \quad \left\{ \sum_{a \in \delta^+(s)} y_a - \sum_{a \in A} w_a \, y_a : (1.5b), (1.5c), (1.5d) \right\} \tag{1.11}$$

The new objective function of problem (1.11) penalizes the use of the interdicted arcs by subtracting their flow from the maximum flow φ. As shown in Cormican et al. [1998], this is *de facto* sufficient to have an optimal follower solution in which the interdicted arcs are not used. We refer to Cormican et al. [1998] for further details on this topic. By dualizing (1.11) and using LP duality, we obtain the following equivalent single-level ILP formulation:

$$\zeta(\text{MFIP}) \;=\; \min \quad \sum_{a \in A} c_a \, \beta_a \tag{1.12a}$$

$$\beta_{uv} + w_{uv} + \alpha_v - \alpha_u \geq 0 \qquad (u,v) \in A, \tag{1.12b}$$

$$\alpha_s - \alpha_t \geq 1 \tag{1.12c}$$

$$\sum_{a \in A} q_a \, w_a \leq \Psi \tag{1.12d}$$

$$\alpha_u \in \{0,1\} \qquad u \in V, \tag{1.12e}$$

$$\beta_a \in \{0,1\} \qquad a \in A, \tag{1.12f}$$

$$w_a \in \{0,1\} \qquad a \in A. \tag{1.12g}$$

It is worth noticing that in any optimal solution, for a given arc $a \in A$, we can have either $\beta_a = 1$ or $w_a = 1$ but not both. This is due to the fact that $c_a > 0, \forall a \in A$. If $w_a = \beta_a = 1$, then the solution obtained by setting $\beta_a = 0$ and keeping other values unchanged is still feasible and does not increase the objective function value. For this reason and due to the nature of constraints (1.12b) and (3.20d), an optimal solution of (1.12) is a cut in the graph G, denoted by $\delta(U^G(\mathbf{w}))$, which depends on an optimal interdiction policy $\mathbf{w}$ (see also Royset and Wood [2007]). More precisely, the

cut $\delta(U^G(\mathbf{w}))$ is given by the arcs $a \in A$ where $\beta_a = 1$ or $w_a = 1$ and it is the union of the set of non-interdicted arcs such that $\beta_a = 1$ and $x_a = 0$ and the set of interdicted arcs such that $w_a = 1$ and $\beta_a = 0$. If a variable α_u is equal to 1, it indicates that vertex u is in the subset $U^G(\mathbf{w})$ containing the source s and if it is equal to 0, it indicates that vertex u is in the subset $V \setminus U^G(\mathbf{w})$ containing the destination t. The objective function (1.12a) minimizes the sum of the capacities of the non-interdicted arcs contained in the cut $\delta(U^G(\mathbf{w}))$. Constraints (1.12b) forces an arc (u, v) to be in the cut if $\alpha_u = 1$ and $\alpha_v = 0$ and accordingly $\beta_{(u,v)} = 1$ or $w_{(u,v)} = 1$. Constraint (1.12c) imposes that $\alpha_s = 1$ and $\alpha_t = 0$, i.e., the source belongs to the set $U^G(\mathbf{w})$ and the destination belongs to the set $V \setminus U^G(\mathbf{w})$. Constraints (1.12b) and (1.12c) extend the classical constraints of the MCP (see Schrijver [2002]), allowing the use of a variable w_a instead of a variable β_a when $\alpha_u = 1$ and $\alpha_v = 0$. For a given arc $(u, v) \in A$, setting the variable $w_{(u,v)}$ to 1 allows to set the variable $\beta_{(u,v)}$ to 0 even if u is in $U^G(\mathbf{w})$ and v is in $V \setminus U^G(\mathbf{w})$, i.e., to reduce the objective function value by $c_{(u,v)}$ but it generates an interdiction cost $q_{(u,v)}$ which consumes a portion of the interdiction budget Ψ.

Any optimal solution $(\mathbf{w}, \boldsymbol{\beta}, \boldsymbol{\alpha})$ of Model (1.12) contains the minimum cut denoted by $\delta(U^{\mathcal{G}_{NI}}(\mathbf{w}))$ in the non-interdicted graph $\mathcal{G}_{NI}(\mathbf{w})$ which is given by the non-interdicted arcs $a \in \mathcal{A}_{NI}(\mathbf{w})$ such that $\beta_a = 1$. This is due to the fact that constraints (1.12b) of Formulation (1.12) can be equivalently rewritten as follows:

$$\beta_{uv} + \alpha_v - \alpha_u \geq 0, \quad \forall\, (u, v) \in \mathcal{A}_{NI}(\mathbf{w}), \tag{1.13}$$

where $\mathcal{A}_{NI}(\mathbf{w})$ are the arcs of $\mathcal{G}_{NI}(\mathbf{w})$. Together with constraints (1.12c), they are the standard MCP constraints for the non-interdicted graph $\mathcal{G}_{NI}(\mathbf{w})$.

Let us consider again the network of Figure 1.5 with a maximum flow equal to 36. Figure 1.7 displays on this network an optimal MFIP solution, with an interdiction cost $\Psi = 11$. For each arc $a \in A$, three associated values are shown in the same configuration as in Figure 3.1. As previously, the interdicted arcs are depicted with dashed blue lines and the thicker line, together with the grey shaded surface, define the minimum cut in the non-interdicted graph. In the depicted optimal MFIP solution, there are two interdicted arcs (s, v_1) and (s, v_2), with a total interdiction cost equal to 11. The optimal value $\zeta(\text{MFIP})$ is equal to 15. This example shows that with an interdiction budget equal to $\Psi = 11$, the maximum flow can be reduced from 36 to 15.

It is worth noticing that the optimal solution represented in Figure 1.7 for the MFIP is contained in the cut $\delta(U^G(\mathbf{w}))$ of G given by the set of arcs $\{(s, v_1), (s, v_2), (s, v_3)\}$, where $U = \{s\}$. We recall that $w_{sv_1} = w_{sv_2} = 1$ and $\beta_{sv_3} = 1$. All other variables of vectors w and β are set to value 0. Moreover, the cut $\delta(U^G(\mathbf{w}))$ is not the minimum cut in G. In fact, in Figure 1.7, the minimum cut is given by the set of arcs $\{(v_1, v_5), (v_4, v_6), (v_4, v_7), (v_3, v_7)\}$. A solution of the interdiction problem consists in removing either one arc from the minimum cut or the arcs (v_1, v_5) and (v_4, v_6). For these combinations of interdicted arcs, the maximum flow remaining in the non-interdicted graph obtained is at least equal to 18, which is greater than 15, the optimal solution value.

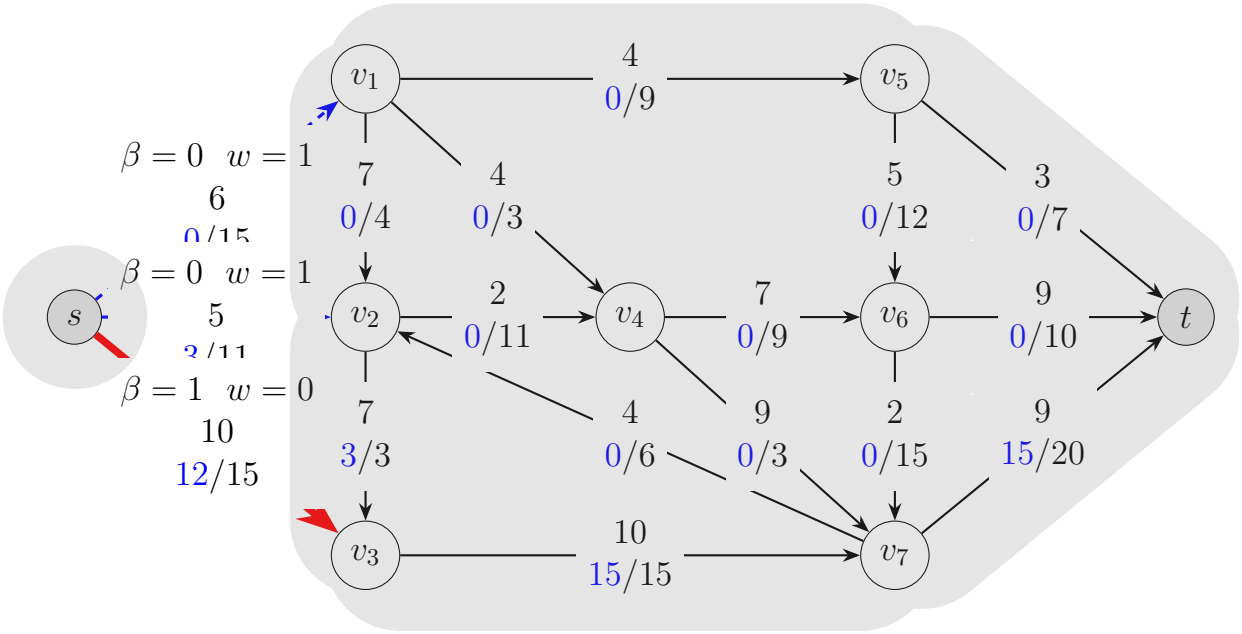


Figure 1.7: An optimal MFIP solution with an interdiction budget $\Psi = 11$.

The multi-commodity flow interdiction problem

We now present a generalization of the maximum flow interdiction problem, called the *multi-commodity flow interdiction problem* (MCFIP). The MCFIP is a bilevel problem where the leader is an interdiction problem and the follower is a multicommodity flow problem. The MCFIP can be formally described as follows. Given a directed graph $G = (V, A)$ and a set of commodities $\mathcal{K}$, the multi-commodity flow interdiction problems aims to find a subset of arcs to be removed from the graph in order to deteriorate as much as possible the objective of the multi-commodity flow problem.

In Lim and Smith [2007b], the authors propose a bilevel model to solve the multi-commodity flow interdiction problem. In the problem treated, the follower is a multi-commodity flow problem solved in a graph $G = (V, A)$, where each arc $a \in A$ has a positive profit value r_a^k for routing one unit of flow for commodity k. The objective of the follower problem is to find a multi-commodity flow with a maximum profit. Therefore, the goal of the MCFIP is to remove a set arcs from the graph in order to minimize the profit of the multicommodity flow remaining in the non-interdicted graph.
Using the notations provided in Section 1.5.2, the bilevel formulation to solve the MCFIP, presented in Lim and Smith [2007b], is given by Model (1.14).

$$\zeta(\text{MCFIP}) \;=\; \min \max \quad \sum_{k \in K} \sum_{a \in A} (y_a^k \; b_k) \; r_a^k \tag{1.14a}$$

$$\sum_{a \in \delta^+(u)} y_a^k - \sum_{a \in \delta^-(u)} y_a^k = \begin{cases} d_k & \text{if } u = s_k \\ 0 & \text{if } u \in \tilde{V_k} \\ -d_k & \text{if } u = t_k \end{cases} \qquad u \in V, k \in \mathcal{K} \tag{1.14b}$$

$$\sum_{k \in K} d_k \cdot y_a^k \leq c_a(1 - w_a) \qquad a \in A, \tag{1.14c}$$

$$y_a^k \in [0,1] \qquad a \in A, k \in \mathcal{K} \tag{1.14d}$$

$$\sum_{a \in A} q_a \cdot w_a \leq R \tag{1.14e}$$

$$w_a \in \{0,1\} \qquad a \in A \tag{1.14f}$$

We denote by $\zeta(\text{MCFIP})$ the optimal solution value of Model (1.14). The objective function (1.14a) represents the profit value of the multi-commodity flow remaining in the non-interdicted graph. Constraints (1.14b) are the *flow conservation* constraints that also ensure that all commodities are satisfied. Constraints (1.14c) are the capacity constraints that take into account the action of the interdiction problem. Constraints (1.14e) is the interdiction budget constraint. Finally, constraints (1.14d) and constraints (1.14f) express the definition set of variables y and w.

In Lim and Smith [2007a], two techniques are presented for solving the MCFIP. The initial approach involves the reformulation of Model (1.14) into a mixed-integer bilinear programming problem. This transformation relies on the fact that, for any solution $\hat{w}$ of the interdiction, the follower problem can be characterized as a multicommodity flow problem. Consequently, by holding the interdiction variables at fixed values, it becomes feasible to substitute the follower's problem with its dual, resulting in a mixed-integer bilinear programming problem. This problem can be tackled using standard linearization techniques, leading to the derivation of a linear mixed-integer programming problem featuring $|V||K| + 2|A|$ continuous variables, $|A|$ binary variables, and $|A|(|K| + 2) + 1$ structural constraints. The second approach is based on a penalty formulation, in which the leader's variables appear only in the objective function, penalizing the follower's use of interdicted arcs. This reformulation asks for determining a large constant representing the penalization term of the interdiction variables in the objective function. Subsequently, as in the previous approach, the penalty formulation is replaced in the master problem by its dual reformulation, resulting in a mixed-integer programming problem featuring $|V||K| + |A|$ continuous variables, $|A|$ binary variables, and $|A||K| + 1$ structural constraints.The two methods are compared through an extensive computational study, demonstrating the effectiveness of the penalty formulation in

contrast to the first approach.

The shortest path interdiction problem

Another network interdiction problem consists of an interdiction problem that applies to the shortest path problem, which is called the *shortest path interdiction problem* (SPIP).

Given a network $G = (V, A)$ where a length $l_a \in \mathbb{R}^+$ is affected to every arc $a \in A$, the shortest path problem, or shortest $s-t$ path problem aims to find an $s-t$ path P that minimizes the sum of lengths of the arcs in $A(P)$. The shortest $s-t$ path problem has been largely studied and many polynomial algorithms have been developed to solve this problem (see Ahuja et al. [1993]). This problem has also been studied in many variants (see Grappe et al. [2023], Benhamiche et al. [2023], Martin et al. [2022]).

Let $\mathbf{q}$ and Ψ be respectively, the interdiction cost vector and the interdiction budget, introduced previously. Let d_a be a delay factor affected to every arc $a \in A$, the *Shortest path interdiction problem* (SPIP) consists in finding a subset of arcs for which the length will be extended by the affected delay, whose does not exceed the budget and so that the shortest path between s and t is maximized. In this case, an interdicted arc refers to an arc for which the length has been extended.
In Israeli and Wood [2002a], the authors propose a bilevel formulation to solve this problem. Let μ_a be a variable affected to each arc $a \in A$ where μ_a takes value 1 if arc a belongs to the shortest path. Otherwise, μ_a takes value 0. Given an interdiction vector $\mathbf{w} \in \mathcal{W}$, the integer linear formulation to solve the SPIP is given by model (1.15).

$$\zeta(\text{SPIP}) \;=\; \max \;\; \vartheta(w) \tag{1.15a}$$

$$\sum_{a \in A} q_a \, w_a \leq \Psi \tag{1.15b}$$

$$w_a \in \{0,1\} \qquad\qquad a \in A, \tag{1.15c}$$

$$\text{where} \quad \vartheta(w) = \min \quad \sum_{a \in A} (l_a + w_a d_a)\mu_a \tag{1.15d}$$

$$\sum_{a \in \delta^+(u)} x_{(u,v)} - \sum_{a \in \delta^-(u)} \mu_{(u,v)} = \begin{cases} 1 & \text{if } u = s \\ 0 & \text{if } u \in \tilde{V} \, u \in V, \\ -1 & \text{if } u = t \end{cases} \tag{1.15e}$$

$$\mu_a \geq 0 \qquad\qquad a \in A. \tag{1.15f}$$

In this bilevel model, the leader problem is given by the first $0-1$ program stated by ((1.15b)-(1.15c)); and the follower problem is given by the inner minimization program stated by ((1.15d)-(1.15f)). As previously, the leader permits to determine the set of

interdicted arcs. The follower aims at finding the shortest path in the non-interdicted graph, i.e, the graph induced by the set of non-interdicted arcs $a \in A$ with $w_a = 0$. The variable $\vartheta(w)$ represents the length of the shortest path in the non-interdicted graph. Constraint (1.15b) guarantees that the interdiction budget is respected and constraints (1.15e) are the path constraints.

A single level reformulation of Model 1.15 has been presented in the article. It is obtained by fixing the interdiction variable values $\mathbf{w}$, take the dual of the follower problem ((1.15d)- (1.15f)) and then release $\mathbf{w}$. This leads to the following MIP:

$$\zeta(\text{SPIP}) = \max_{\mathbf{w}\in\{0,1\}^m} \quad \pi_t - \pi_s \tag{1.16a}$$

$$\pi_v - \pi_u - d_a\, w_a \leq 0 \qquad a = (u,v) \in A, \tag{1.16b}$$

$$\pi_s = 0 \tag{1.16c}$$

$$\sum_{a\in A} q_a\, w_a \leq \Psi \tag{1.16d}$$

$$\pi_u \text{ unrestricted} \qquad u \in V \tag{1.16e}$$

where $\boldsymbol{\pi}$ are the dual variables associated with Constraints (1.15e).

However, in Israeli and Wood [2002b], the authors solves the bilevel problem using decomposition algorithms. More precisely, the algorithm is based on Benders decomposition for which integer cutting planes are added to the master problem. A second approach consists in a reformulation of the master problem into a set-covering master problem.

1.6.3 Blocker problems

A blocker problem is a bi-level optimization problem that consists in performing a set of operations in the underlying problem to destruct its structure. The objective of this problem is to minimize the cost of the operations. In this work, we study the blocker problem applied to the maximum flow problem and to the multi-commodity flow problem.

The notion of interdiction is very close to the blocker notion. However, it is important to keep the two processes apart. The blocker problem aims to destruct the sub-problem with a minimum cost while the interdiction problem tries to deteriorate as much as possible the sub-problem, respecting the budget constraint.

In what follows, we present a specific blocker problem that applies to the shortest path problem. This problem is denoted by the most vital vertices for the shortest path problem (MVVSPP) and it has been studied in Magnouche and Martin [2020].

The most vital vertices for the shortest path problem

In the most vital vertices for the shortest path problem, the blocker problem aims to remove from a network a set of vertices with a minimum cost, called *blocker cost*. This blocker problem applies to a shortest path problem.

More formally, given a directed graph $G = (V, A)$ with a source $s \in V$ and a destination $t \in V$, where every vertex $v \in V$ has a blocker cost $b_v \in \mathbb{Z}_+$, and every arc $a \in A$ has a length $l_a \in \mathbb{Z}_+$, the MVVSPP consists in finding a set of vertices with a minimum total cost whose removal involve that the shortest path remaining in the graph is greater than or equal to a target length value denoted by ξ.

Let $\mathbf{z}$ be a vector of binary variables associated with the set of vertices V of a graph G, each variable encoding whether the corresponding vertex is removed from the graph or not. A natural formulation for the MVVSPP reads as follows:

$$\min \quad \sum_{v \in V} b_v z_v \tag{1.17a}$$

$$\sum_{v \in P} z_v \geq 1, \qquad p \in \mathcal{P}_l, \tag{1.17b}$$

$$z_v \in \{0, 1\} \qquad v \in V, \tag{1.17c}$$

where $\mathcal{P}_l$ is a collection of paths from s to t with a length less than or equal to the target length value ξ. It is worth noticing that this formulation has an exponential number of constraints that consists in removing from every feasible path, i.e., every path with a length less than or equal to ξ, at least one vertex.

We denote by $P_{MVVSPP}(G)$ the polytope of this problem, i.e,

$$P_{MVVSPP}(G) = conv(\{\mathbf{z} \in \{0, 1\}^m \quad : \quad \mathbf{z} \text{ satisfies (1.17)}\}. \tag{1.18}$$

In Magnouche and Martin [2020], the authors conducted a polyhedral analysis to enhance Model (1.17). More precisely, they have shown that the polytope $P_{MVVSPP}(G)$ is integer.

Chapter 2

Network flow blocker problems and their applications in Telecommunication Networks

Contents

In this chapter, our focus lies in exploring the practical applications of network flow blocker problems, with a specific emphasis on its relevance within the Telecommunication industry. To this end, we first give a complete description of a telecommunication network, including its architecture and the routing process. Then, our initial application centers on the evaluation of telecommunication network resilience. Following this, we delve into the realm of network monitoring, and more in particular, the placement of approximate monitoring data structures called sketches. Within this context, we study the Flow-Sketch assignment problem, for which we develop a bilevel approach and a greedy algorithm to effectively tackle this problem.

2.1 Resilience analysis

Modern Telecommunication Networks (see Gutierrez-Estevez et al. [2019]) are extremely complex systems in which hard combinatorial optimization problems must be effectively solved in a short computing time. Indeed, the emergence of 5G enables a broad set of diverse services characterized by complex and potentially conflicting demands (see Martin et al. [2021], Krolikowski et al. [2021a], Huin et al. [2023]). In this section, we delve into a comprehensive examination of the management and structure of telecommunication networks in the context of modern real-world scenarios. We then focus on the routing process, explaining the mechanisms that govern the efficient transmission of data within these networks. Finally, we address the potential anomalies that may arise in these networks. Indeed, unexpected disruptions, anomalies, or deviations from anticipated standards can affect the reliability and stability of telecommunication services. By identifying and understanding these anomalies, solutions can be developed to enhance the resilience of modern telecommunication networks.

2.1.1 Networks architecture

Telecommunication networks are dynamic and complex systems composed of a large number of devices such as switches or routers and numerous types of *Service Functions* (SFs) such as firewalls, deep packet inspections (DPIs), web proxies, media gateways, etc. These SFs are used as middleboxes on the Service Provided Networks, operating within the network infrastructure to perform specific tasks like routing, filtering, or processing data. Positioned strategically between the sender and the receiver, they deliver a range of services without compromising the integrity of end-to-end communication. SFs are deeply linked to the efficient satisfaction of the demand and their optimization is strongly connected to demand peaks and anomalies. Indeed, these SFs play a crucial role in ensuring the seamless operation of telecommunication networks by adjusting to diverse demands and addressing potential irregularities to achieve optimal performance.

Software Defined Networking (SDN) is an innovative paradigm in which the overall behavior of the network is directed by a central software program known as a controller. In SDN, network devices simply forward packets, with the control logic embedded in the controller. More precisely, SDN has two key components: the controller and the switches. The SDN controller is responsible for managing the entire network, while networking switches operate based on instructions deployed through the SDN controller. Unlike traditional networks, where the entire system requires reconfiguration for upgrades, only the software needs updating in SDN. Moreover, in SDNs, Network Function Virtualization enables the virtualization of Service Functions (SFs). This disassociation of SFs from the physical elements of the network allows them to be installed as software. This ensures a centralized control of network resources in which the controller has visibility over the entire network and has a complete view of the network topology.

Figures 2.1 and 2.2 illustrate the difference between the traditional networking architecture and SDN architecture. Note that a router's primary role is to facilitate the routing of data between different networks while a switch is employed to connect devices within

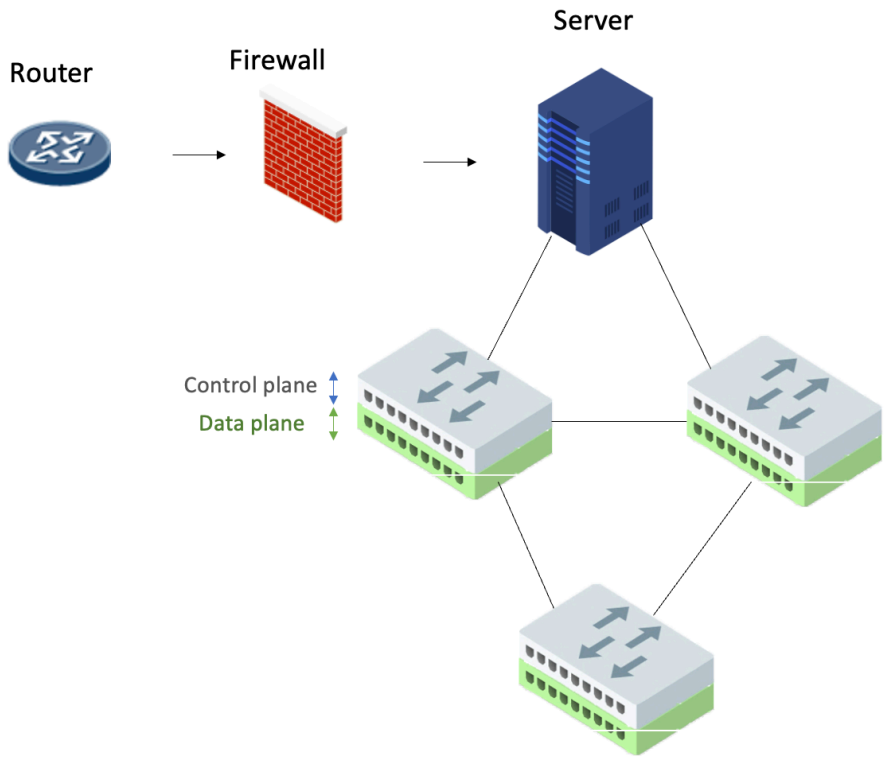


Figure 2.1: Traditional network architecture

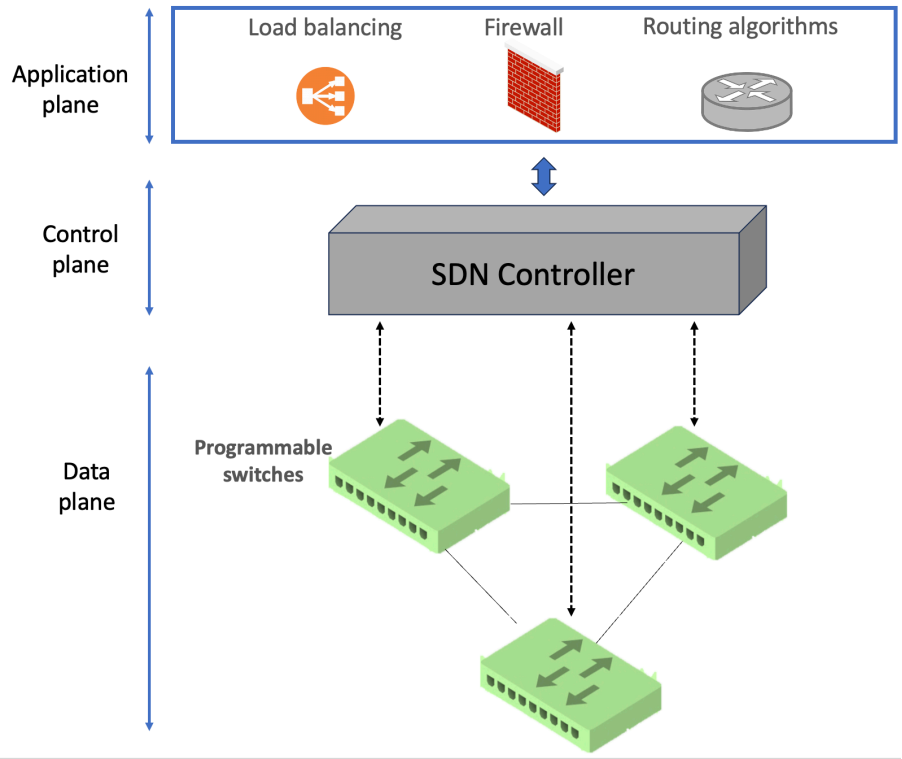


Figure 2.2: SDN architecture

the same local area network (LAN). In SDN, there is a distinct separation of the application plane, data plane, and control plane. The application plane is responsible for the network's specific functionalities and services, such as routing decisions and security policies. The data plane handles the actual transmission of data packets within the

network, focusing on packet forwarding and switching. The control plane manages the overall network by making decisions based on the information received from the data plane and application plane. It determines how data packets should be forwarded and updates the network accordingly. By decoupling these planes, SDN allows for a more flexible and programmable network architecture, enhancing adaptability and management capabilities.

2.1.2 Routing in telecommunication networks

A telecommunication network is a complex system that allows data to be transmitted over long distances. Routers and switches play a pivotal role by directing data packets efficiently from their source to their destination. These data packets are transmitted along transmission links, which can be physical cables or wireless connections, at high speeds.

Routing refers to the process by which routers determine the optimal path for each data packet to traverse from its source to its designated destination. This task involves complex decision-making. Routers evaluate multiple factors, including network congestion, available bandwidth and latency, to make routing decisions in real-time, ensuring that data packets are efficiently guided along the transmission links.

Segment routing (see Polverini et al. [2018]) is a new routing concept that can be used to steer traffic from a source along any arbitrary path in the network. It is often based on precomputing several paths for each path in the network. Also, it allows the operators to efficiently load balance traffic while meeting low latency and failure tolerance performance targets.

Telecommunication networks find a natural representation as graphs. In these graphs, vertices typically represent the network elements such as routers, while edges represent the transmission links connecting them. This graph-based representation simplifies the analysis and management of the network. By studying the graph, network administrators can gain insights into its structure, identify potential bottlenecks, and optimize routing paths. It allows for a clear visualization of how data flows through the network, making it an invaluable tool for ensuring efficient and reliable communication in modern telecommunications systems.

o

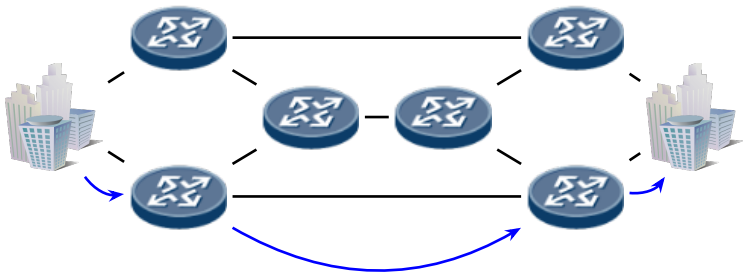


Figure 2.3: Routing process

Figure 2.3 depicts a graph representation of a routing process between two terminals, passing through routers. The routing is illustrated by curved blue arcs.

2.1.3 Anomalies in telecommunication networks

Anomalies in telecommunication networks, whether arising on routers or transmission links, can significantly disrupt the smooth flow of data and compromise the reliability of communication systems. Routers, as critical network components, are susceptible to various anomalies, including hardware failures, software glitches, or configuration errors. These anomalies can lead to routing loops, where data packets circulate endlessly within the network, causing congestion and packet loss. Transmission links, whether physical or wireless, are equally susceptible to anomalies such as signal interference, cable damage, or bandwidth saturation. We refer the interested reader to Junior et al. [2019] for further details on telecommunication network anomalies. These issues can result in data degradation, increased latency, or even complete link failures.

Identifying and addressing these anomalies is of vital importance for network administrators. To tackle this issue, many tools are employed such as monitoring tools, anomaly detection algorithms, and redundancy mechanisms. Anomalies in routers or transmission links can disrupt the overall network performance. Therefore, proactive maintenance and robust contingency plans are essential to ensure uninterrupted and reliable telecommunication services.

2.1.4 Resilience analysis with network flow blocker

The key purpose behind investigating network flow blocker problems lies in evaluating the performance of a telecommunication network, particularly to analyze its resilience against anomalies.

A *network flow blocker problem* refers to a blocker problem applied to a network flow problem. Here, the network flow problem is a representation of the telecommunication network, essentially defining its structure and operational characteristics. The blocker problem represents the potential anomalies that can manifest within the network, subsequently influencing the routing process.

Telecommunication Networks are often faced with the question of identifying the most vital (also called most vulnerable or most critical) part of the network, which corresponds to a subset of vertices (or edges) of limited size, whose malfunctioning prevents the functionality of the network as a whole. Depending on the crucial property that needs to be maintained (or achieved) in the network, different vertices may be perceived as the most important ones. In some applications, one might ask for the most critical vertices that may affect or destroy connectivity of the network, see, e.g. Lalou et al. [2018]. Moreover, as explained previously, using Software Defined Networks (SDNs), SFs can be disassociated from the physical infrastructure and operated as software, allowing a centralized control of network resources. In this context, studying network resilience allows a better understanding of the behavior of the telecommunication networks and their control especially taking into account the important decisions linked

to demand anomalies and Services Functions (SFs). In addition, the correct functioning of telecommunication networks frequently depends on a small number of important vertices whose malfunctioning can significantly degrade the performance of the whole network.

In this thesis, we focus on two classes of network flow problems. The first one consists of a single commodity that aims to maximize the amount of data from a source to a destination, as illustrated in Figure 2.3. The associated network flow blocker problem is called the *maximum flow blocker problem* (MFBP). The second one consists of multi-commodity flow problems arising in telecommunication networks with segment routing concepts and complex demand satisfaction constraints. More precisely, we are interested in demands having different shapes of traffic and predicted with machine learning methods. The associated network flow blocker problem is called *multi-commodity flow blocker problem* (MCFBP).

Let us illustrate how a blocker problem can assess the resilience of a network with a single commodity, specifically the maximum flow blocker problem. In telecommunication networks, the flow routed on an arc represents the amount of data sent from the source to the destination and passing through that arc. Malfunctioning arcs can be modeled by reducing the capacity of the arc to 0 or more generally by removing the arc from the graph. These situations are due to anomalies, failures, or packet loss caused by congestion. The MFBP corresponds to the blocker variant of the maximum flow problem in which each arc is also given a *blocker cost*. It consists of finding a minimum-cost subset of arcs to be removed from the graph in such a way that the maximum flow value in the remaining graph is no larger than a given threshold. In this context, MFBP optimal solutions allow analyzing the network resilience in case of malfunctioning arcs.

In the case where all blocker costs are equal to one, a network is said to be *resilient* to f simultaneous arc failures, with respect to a target flow Φ, if after the removal of any set of at most f arcs, there exists a flow of value larger than Φ. An optimal MFBP solution is a set of arcs of minimum cardinality $\zeta(G)$ such that the network is not resilient to $\zeta(G)$ simultaneous arc failures with respect to a target flow Φ. By considering $f = \zeta(G) - 1$, since $\zeta(G)$ is the minimum size of the set of arcs, removing any set of arcs of size at most $\zeta(G) - 1$ (maximum simultaneous malfunctioning arcs) ensures that the maximum flow value in the remaining graph is larger than Φ. Let us consider a network represented by the graph shown in Figure 2.4, with all blocker costs equal to one. We report on each arc its blocker cost shown in red above the flow of the arc in the remaining graph separated by "/" with the arc capacity. The threshold, denoted by Φ is set to 10. The optimal solution is represented with dashed blue arcs, (v_6, t) and (v_7, t), and its value is equal to 2. The consequence of removing these arcs results in a graph containing a maximum flow value of 7. Accordingly, for any single malfunctioning arc, the remaining graph has a maximum flow value larger than 10. The MFBP solution is represented in Figure 2.4.

Moreover, always in telecommunication networks with general blocker costs, MFBP solutions determine the minimum-cost subset of arcs to monitor all flows of value strictly larger than Φ. More precisely, for each subset of arcs guaranteeing a flow strictly greater than Φ, at least one of them is monitored. The blocker costs in this case represent the

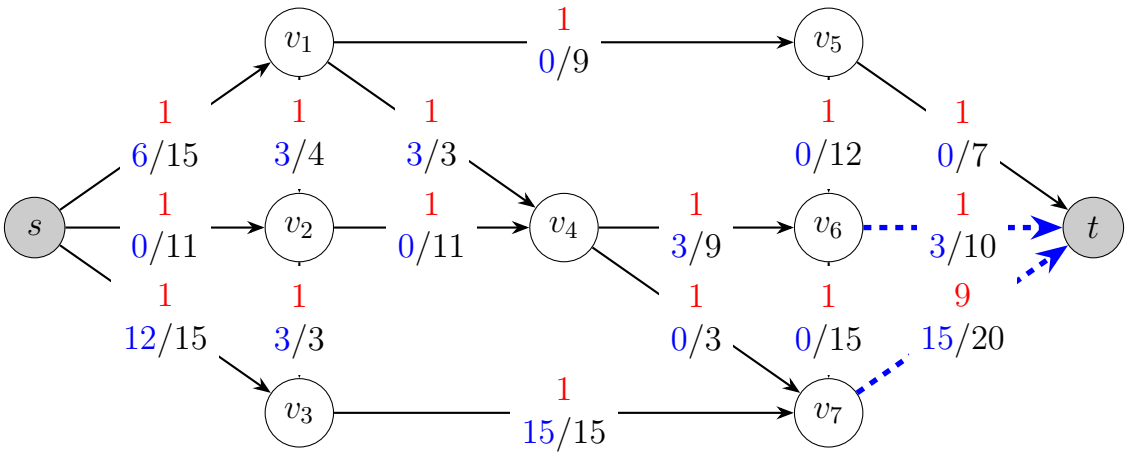


Figure 2.4: An optimal MFBP solution with a threshold equal to 10.

costs of installing the monitor devices on the arcs. We refer the interested reader to Ghafir et al. [2016] for further details on network monitoring.

The multi-commodity flow blocker problem, in the context of analyzing the resilience of telecommunication networks, follows a comparable line of reasoning. The distinction between the MFBP and the MCFBP depends mostly on the characteristics of the networks. For instance, in the context of a network with multiple commodities, it may pursue various objectives to optimize flow routing, such as maximizing the number of satisfied commodities, minimizing the total routing cost, or maximizing the flow between each commodity. Moreover, the definition of network functionality depends on several factors and can be aligned with specific applications of the multi-commodity flow blocker problem (MCFBP).

In what follows, we provide a detailed example illustrating the application of a multi-commodity flow blocker problem, with a specific focus on anomalies occurring on network vertices. This problem is referred to as V-MCFBP. In real-world applications, networks often accommodate multiple commodities, each necessitating the transfer of specific data between distinct terminal pairs. Routers, therefore, assume a crucial role in managing and routing these flows, thereby ensuring efficient network-wide communication. In this context, optimal solutions of the MCFBP offer valuable insights into assessing network resilience in the face of router malfunctions. The disruption caused by malfunctioning routers can be effectively modeled by the removal of corresponding vertices from the network graph. Such malfunctions might occur due to a variety of reasons, including deliberate attacks by malicious entities, network congestion, hardware failures, or equipment damages. We refer the interested reader to G. Fernandes [2019] and Li et al. [2018] for further details on router anomalies and how to diagnose them. As for the MFBP, in a V-MCFBP where blocker costs are set to 1 on the vertices of the graph, the solution represents the maximum allowable number of router failures within the network, while maintaining a routing profit exceeding the specified threshold. Alternatively, different blocker cost values can represent the installation of monitoring devices on the routers.

In the next section, we broaden the primary goal of the thesis, aiming to analyze the

resilience of a network. For this purpose, we explore an additional application specifically designed for telecommunication networks. As we strive to enhance overall network performance and implement effective network monitoring strategies, our focus turns to another closely related issue, known as the sketch assignment problem.

2.2 Network-wide sketching

Internet Service Providers (ISPs) often need to monitor the network traffic to perform some tasks based on observed metrics, such as identifying the longest flows. This process involves recognizing Distributed Denial of Service (DDoS) attacks within a network. DDoS detection mechanisms are implemented to identify patterns or anomalies in network traffic that may indicate such an attack. Once detected, appropriate measures can be taken to mitigate and counteract the impact of the attack, helping to ensure the availability and normal operation of the targeted services.

Due to the hardware limitations of network devices, such measurements are nowadays performed using an approximate data structure known as *sketches*. These sketches offer approximate per-flow statistics at a sublinear memory cost and with a certain error bound. Consequently, the strategic placement of these sketches is a complex challenge, requiring a comprehensive understanding of network topology and traffic. The goal of strategically placing these sketches is to achieve effective and balanced monitoring without imposing unnecessary resource requirements and while minimizing potential errors.

In this context, this section proposes a method to tackle the problem of network-wide monitoring, for which statistics are collected jointly by multiple network devices and aggregated to obtain measurements for an entire network perspective.

2.2.1 Network-wide monitoring and related works

Network monitoring and measurement play integral roles in effective network management. The capability to monitor network traffic in real-time and at a broad scale is essential for a variety of tasks, including traffic engineering and the identification of attacks and anomalies (see Zhang [2013]). Each such management task requires accurate and timely statistics on diverse metrics, such as flow size distribution, entropy measures (Lall et al. [2006]), and the identification of changes in traffic patterns. A first technique to estimate these metrics relies on generic flow monitoring, in which traffic measurements are performed on flows that are collected by network devices including routers and switches. Note that a flow is defined as a set of packets matching some attributes (e.g., IP address, port, protocol). Accordingly, flow monitoring is typically performed by means of packet sampling, which can be less accurate for fine-grained metrics, as demonstrated in many studies (see Pescapé et al. [2010]). This leads to an alternative technique based on sketching algorithms. Moreover, the increased programmability of network elements (e.g., programmable switches in SDNs) enables significant progress in sketch-based monitoring techniques, leading to the achievement of precise and detailed network-wide monitoring.

Definition of sketch-based monitoring

Sketch-based monitoring techniques rely on the usage of approximate data structures to represent network traffic in a compact matter; such data structures are then used to extract traffic metrics with certain error bounds. We refer the interested reader to Cormode [2013] and Cormode [2017] for further details on sketch-based data structures.

A sketch is a synopsis data structure typically used in algorithms that process data streams, which are also known as data streaming algorithms. Sketches are designed to provide concise summaries of large volumes of data with a limited amount of memory, making them particularly useful for scenarios where maintaining the entirety of the data is impractical or infeasible. Computations can then be applied to this compact summary instead of the original data stream. Sketches typically involve the use of hash functions to map elements of the data to a fixed-size array or set of counters.

In Figure 2.5, we provide an illustrative example of the Count-Min sketch (refer to Cormode [2009]). The Count-Min sketch is a data structure represented as an array of counters, organized into d hash tables (referred to as h_1, h_2, h_3), each of the size of w. Each hash function maps has a certain incoming flow (e.g., specific 5-tuple) to a counter of the hash table (sketch row). This structure is particularly useful for approximating frequency counts of elements in a data stream. At update time, i.e., whenever a packet of a flow is received, every counter associated with that flow is incremented by one. At query time, the size of a flow can be estimated with the minimum counter among those the flow is mapped to in the sketch. Within the context of the sketch, each row in the array is associated with a specific hash function selected from a universal class of hash functions (as detailed in Dietzfelbinger [2018]). These hash functions play a crucial role in mapping elements to the respective counters, facilitating efficient and space-conscious approximations in scenarios like streaming data analysis.

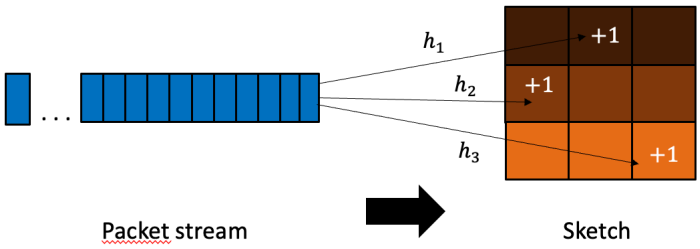


Figure 2.5: Count Min Sketch representaton

Figure 2.6 provides a visual representation of how sketches are integrated and utilized within the SDN framework.

Each sketch features a certain approximation error or estimation error, that depends on the sketch size and the overall number of packets counted. More precisely, as outlined earlier, a sketch generally demands $w \times d$ memory units, where w denotes the sketch width, and d represents its depth. For the purpose of our analysis, we assume a fixed depth for all sketches, focusing on the width parameter, denoted as w. Consequently, the memory capacity of a device $v \in V$ is characterized by its maximum supported

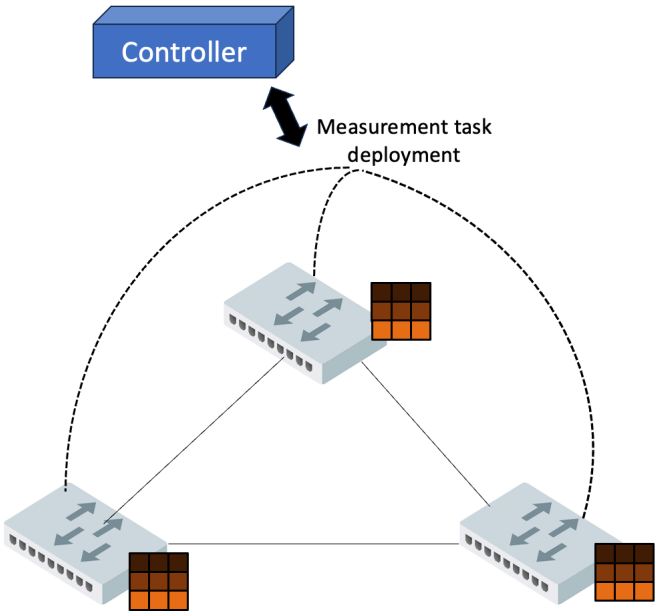


Figure 2.6: Sketches in SDN architecture

width, referred to as W_v. Accordingly, we denote by ε_v the sketch relative error that depends on the width w of the sketch.

It is worth noticing that sketches perform measurements in a distributed fashion, meaning that data processing is spread across multiple devices. This helps reduce collisions on the single device, increase overall capacity and reduce measurement error.

In Figure 2.7, we show a simple network with four terminals (S_1, T_1, S_2, T_2) and three sketches sk_1, sk_2, sk_3. We compare the measurement error using single devices or multiple devices. On each flow, the error is bounded by the quantity $\varepsilon \times N_{S,T}$ where S is the source, T the destination and $N_{S,T}$ is the number of packets going from S to T. Using single device, the overall error is computed as follows: $\varepsilon \times 4(N_{S_1,T_1} + N_{S_1,T_2} + N_{S_2,T_1} + N_{S_2,T_2})$, while using multiple devices, the overall error is computed as $\varepsilon \times 2(N_{S_1,T_1} + N_{S_1,T_2}) + \varepsilon \times 2(N_{S_2,T_1} + N_{S_2,T_2})$.

The trade-off in using sketch data structures is that they offer approximate results with controlled error bounds, making them well-suited for applications where the computational cost or memory requirements of exact solutions are prohibitive. Moreover, the flexibility and resource-efficient nature of sketches enables them to be deployed everywhere, whether in edge computing, IoT devices, or distributed networks.

Prior art

Sketch monitoring has received lots of interest recently, focusing on the objective of measuring and summarizing complex data structures within flows. In this context, we introduce three frameworks from the literature that address this goal. One of these frameworks, namely *DISCO*, (see Bruschi et al. [2020]), is specialized in collecting measures from devices along a single path. These measures are then processed on a controller that acts like a single sketch. It is worth noticing that the goal of this framework is to

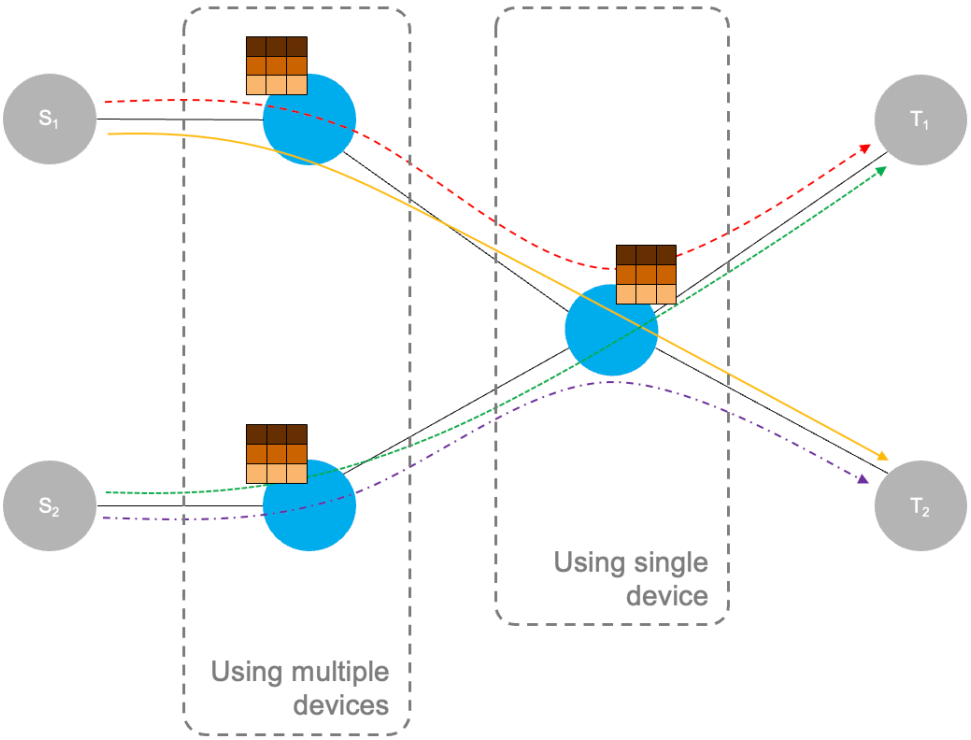


Figure 2.7: Illustration of sketch placement strategies

reduce the error by exploiting counters on multiple devices. However, it only manages a single path. In contrast to *DISCO*, another framework named *UnivMon* is designed in Liu et al. [2016] to accommodate multiple paths, seeking a generic measurement algorithm. This framework, as the framework *HeteroSketch* presented in Agarwal et al. [2022], does not minimize the measurement error. Indeed, *HeteroSketch* presents an automated solution for sketch placement that focuses on the heterogeneity in devices. This approach combines an automated sketch-based device profiler with a hierarchical optimizer designed for efficient flow-device assignment.

Contributions and solution approach

As mentioned earlier, implementing network-wide monitoring through sketches presents various challenges that require careful consideration. One key challenge involves ensuring the proper assignment of existing flows, such as distinct TCP/IP communications, to the various sketches distributed across network devices. The second challenge relies on the fact that a commodity may use multiple paths. In this context, maintaining transparency in network routing is critical. In other words, the monitoring system should not depend on specific routing paths and must seamlessly support multi-path routing configurations. Finally, the optimization of flow assignment is essential to accommodate fluctuating traffic trends effectively. Adapting the assignment strategy in response to changing traffic patterns ensures that the distribution of measurement tasks remains optimized.

However, as mentioned previously, the use of a sketch features an approximation error. Therefore, monitoring all devices on every path increases the approximation error, espe-

cially when dealing with huge amounts of flow. In addition, ignoring some paths leads to partial-coverage error. Finding a balance between these two aspects is not trivial, especially for a significant volume of traffic. Accordingly, we propose in this chapter a monitoring system based on a method for distributed sketch assignment with the goal of minimizing the total approximation error.

Our approach facilitates dynamic optimization of the assignment between network flows and monitoring devices taking into account traffic fluctuations and supporting previously unknown flows that possibly span across multiple network paths. This adaptive approach ensures efficiency and accuracy despite the constantly changing nature of network traffic. Unlike prior art, this solution approach designs a system that can dynamically schedule the assignments between network flows and devices. This dynamic scheduling allows to properly benefit from the distributed monitoring system, e.g., by constantly seeking the optimal assignment that minimizes measurement error. A patent has been granted for this invention to Castellano et al. [2023] and is currently in the process of publication.

The overall idea of our solution approach involves strategically placing a set of sketches within the network, ensuring that each flow is assigned to the most appropriate device along each of its paths. As previously explained, the placement of sketches introduces an error known as sketch estimation error. This error is characterized by the estimation error on each sketch per flow that depends on the total number of packets counted by the sketch. In order to reduce in-sketch overlaps and enhance accuracy, it might be convenient to deliberately ignore certain paths occasionally. Consequently, on such paths, a flow remains unassigned to any device, resulting in a new error known as uncovered paths error. Consequently, achieving an optimal placement requires a balance between minimizing sketch estimation error and uncovered paths error. The challenge lies in finding a placement strategy that effectively navigates this trade-off.

To implement this solution approach we introduce the *Flow-Sketch Assignment problem* (FSAP), which constitutes the core challenge in efficiently managing the placement of sketches within the network. After formally describing the problem, we present two different models to address the FSAP. The first model is an Integer Linear Programming (ILP) formulation. The second model is a bilevel formulation, offering an alternative perspective. We then delve into a detailed discussion of the resolution methods for both models.

2.2.2 The Flow-Sketch assignment problem

Notations

Let $G = (V, E)$ be a directed graph representing the network. V is the set of n vertices and E is the set of m edges. Let F be a set of Origin-Destination (OD) pairs, where each pair $f \in F$ is associated with a source node $s_f \in V$ and a destination node $t_f \in V$. Additionally, for each OD-pair, we assign a flow size parameter n_f denoting the number of packets associated with that specific pair. We denote by $n(v)$ the total number of packets counted by node v.

It is worth noticing that, based on the network topology G, each OD-pair may traverse multiple paths. Moreover, as explained earlier, some flows on paths will be ignored to prevent an increase in error that would result from counting them. Therefore, we suppose that the flow sizes n_f are initially unknown, and our objective is to estimate these sizes for each OD-pair $f \in F$. Therefore, we introduce the set of commodities K where for each $k \in K$, s_k is the source and t_k is the destination. A commodity refers to a source and a destination responsible for transferring data. Note that the source s_k (resp. the destination t_k) corresponds to the source s_f (resp. the destination t_f) for all OD-pairs $f \in F$. For every commodity $k \in K$, let $\mathcal{P}(k)$ be a function returning all possible paths from s_k to t_k and $\mathcal{V}(P)$ be a function returning all nodes across path P. We recall that ε_v corresponds to the sketch relative error.

In the next section, we present a particular case of the Sketch Assignment Problem, called the *The Min Error Sketch Assignment problem* for which the flow size is fixed for all OD-pairs.

The Min Error Sketch Assignment problem

The Min Error Sketch Assignment problem consists of finding a placement scheme for sketches such that all packets from existing OD-pairs are covered (i.e., there is a dedicated sketch for each of their possible paths), and the overall estimation error is minimized.

Note that multiple tasks may be necessary for each flow. This means that every packet within the flow must be seen by multiple sketches, and the resources of a node may be divided among these tasks. However, in our study, we focus on scenarios where there is only one task per flow. Therefore, every node $v \in V$ allocates a single sketch using all the available memory W_v. In this particular problem, our objective is to determine the optimal assignment of flows to nodes.

Let us introduce a vector $\mathbf{x}$ of binary variables, each variable $x_{f,v}$ is associated with a flow $f \in F$ and a node $v \in V$ and it takes value 1 if and only if flow f is assigned to a sketch on node v i.e., there are packets from the OD-pair f that are covered by a sketch on node v.

The Min Error Sketch Assignment problem aims to find the assignment $\mathbf{x}$ that minimizes the overall estimation error bound computed as follows:

$$\sum_{f \in F} \sum_{v \in V} x_{f,v}\ \varepsilon_v\ n(v)$$

As mentioned earlier, the error term ϵ_v is associated with the sketch dimension. Moreover, it is worth noticing that when computing the overall estimation error, this relative error ϵ_v is multiplied by the number of packets covered on node v.

We now present a graphical example to illustrate the Min Error Sketch Assignment Problem thanks to the graph represented in Figure 2.8 with 7 vertices and 7 arcs. We consider the following set of OD-pairs: $\{(1,6),(1,7),(2,6),(2,7)\}$. In the figure, we

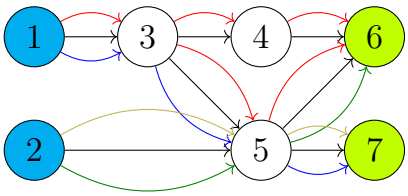


Figure 2.8: Example graph for the Min Error Sketch Assignment problem

illustrate the paths corresponding to each OD-pair using distinct colors: red for the first OD-pair, blue for the second, green for the third, and yellow for the fourth.

The goal of the Min Error Sketch Assignment problem is to assign each OD-pair to a set of nodes that should monitor it. Based on the network topology, each OD-pair may be assigned to different nodes. For instance, the first OD-pair $(1, 6)$ can be either entirely assigned to node 3, or split across nodes 5 and 4. The second OD-pair $(1, 7)$ can be entirely assigned to node 3 or to node 5. The third and fourth OD-pairs $(2, 6)$ and $(2, 7)$ can be assigned to node 5. These assignments allow to count every packet exactly once.

We now analyze two possible solutions.

Solution 1. This solution consists of monitoring the first two OD-pairs $(1, 6)$ and $(1, 7)$ on node 3. The last two OD-pairs $(2, 6)$ and $(2, 7)$ are monitored on node 5. Therefore, we have:

$$x_{1,3} = x_{2,3} = x_{3,5} = x_{4,5} = 1,$$

while $x_{f,v} = 0$ for any other OD-pair $f \in F$ and node $v \in V$.

We assume that $\varepsilon_v = \varepsilon, \forall v \in V$. The overall error is bounded by:

$$\begin{aligned}
x_{1,3}\,\varepsilon\,n(3) + x_{1,4}\,\varepsilon\,n(4) + x_{1,5}\,\varepsilon\,n(5) + \\
x_{2,3}\,\varepsilon\,n(3) + x_{2,4}\,\varepsilon\,n(4) + x_{2,5}\,\varepsilon\,n(5) + \\
x_{3,3}\,\varepsilon\,n(3) + x_{3,4}\,\varepsilon\,n(4) + x_{3,5}\,\varepsilon\,n(5) + \\
x_{4,3}\,\varepsilon\,n(3) + x_{4,4}\,\varepsilon\,n(4) + x_{4,5}\,\varepsilon\,n(5) = \\
\varepsilon\,n(3) + \varepsilon\,n(3) + \varepsilon\,n(5) + \varepsilon\,n(5) = \\
2\,\varepsilon\,n(3) + 2\,\varepsilon\,n(5) = 2\,\varepsilon\,(n_1 + n_2) + 2\,\varepsilon\,(n_3 + n_4) = 2\,\varepsilon\,N
\end{aligned}$$

Solution 2. This solution consists in monitoring the first OD-pair $(1, 6)$ on on nodes 4 and 5, the second OD-pair $(1, 7)$ on node 3, and the last two pairs on node 5. Therefore, we have:

$$x_{1,4} = x_{1,5} = x_{2,3} = x_{3,5} = x_{4,5} = 1,$$

while $x_{f,v} = 0$ for any other OD-pair $f \in F$ and node $v \in V$.

In this case, the overall error is given by:

$$\begin{aligned}
x_{1,3}\,\varepsilon\,n(3) + x_{1,4}\,\varepsilon\,n(4) + x_{1,5}\,\varepsilon\,n(5) + \\
x_{2,3}\,\varepsilon\,n(3) + x_{2,4}\,\varepsilon\,n(4) + x_{2,5}\,\varepsilon\,n(5) + \\
x_{3,3}\,\varepsilon\,n(3) + x_{3,4}\,\varepsilon\,n(4) + x_{3,5}\,\varepsilon\,n(5) + \\
x_{4,3}\,\varepsilon\,n(3) + x_{4,4}\,\varepsilon\,n(4) + x_{4,5}\,\varepsilon\,n(5) = \\
\varepsilon\,n(4) + \varepsilon\,n(5) + \varepsilon\,n(3) + \varepsilon\,n(5) + \varepsilon\,n(5) = \\
\varepsilon\,n(3) + \varepsilon\,n(4) + 3\,\varepsilon\,n(5) = \\
\varepsilon\,n_2 + \varepsilon\,n_1(4) + 3\,\varepsilon\,(n_1(5) + n_3 + n_4) = 2\,\varepsilon\,(N/2 + n_3 + n_4 + n_1(5)),
\end{aligned}$$

where $n_f(v)$ is the flow of the OD-pair f routed on node v, $\forall f \in F, v \in V$.

It is worth noticing that these two distinct solutions result in different estimation errors.

In the next section, we propose an integer linear programming (ILP) formulation to solve the sketch assignment problem for which some paths of a flow will be ignored, i.e., all packets will not be covered.

2.2.3 An ILP formulation for the FSAP

In this section, we first focus on a version of the Flow-Skecth assignment problem (FSAP) with the objective of minimizing the estimation error. We introduce an ILP model to address this problem. We then extend the ILP model to optimize the trade-off between the estimation error associated with sketches and the error originating from uncovered paths.

Flow-Sketch Assignment problem

Let K be a set of commodities where for each $k \in K$, s_k is the source and t_k is the destination. For each commodity $k \in K$, we introduce a parameter Ψ_k, corresponding to a flow-size threshold. The goal of the flow-sketch assignment problem (FSAP) is to find the smallest estimation error such that for each $k \in K$ all flows between s_k and t_k of size greater than Ψ_k are covered by a sketch.

Let us introduce a vector $\mathbf{x} \in \{0,1\}^{|K|\times|V|}$ of $|K| \times |V|$ binary variables, each variable $x_{k,v}$ equals to 1 if a sketch is installed on v to monitor the flow between s_k and t_k and 0, otherwise.

A model for the FSAP reads as follows:

$$\min \quad \sum_{k\in K}\sum_{v\in V} \varepsilon_v C_k(v) x_{k,v} \sum_{k'\in K} x_{k',v} \tag{2.1a}$$

$$\sum_{v\in\mathcal{V}(F)} x_{k,v} \geq 1 \qquad \forall k \in K,\ F \in \mathcal{F}_k(\Psi_k) \tag{2.1b}$$

$$x_{k,v} \in \{0,1\} \qquad \forall k \in K,\ v \in V, \tag{2.1c}$$

where $\mathcal{V}(F)$ is a function returning the set of vertices covered by at least one unity of flow F and $\mathcal{F}_k(\Psi_k)$ is the set of flows between s_k and t_k with a size greater than Ψ_k and $C_k(v)$ is the node capacity, i.e., $C_k(v)$ gives the maximum observable load (or traffic flow) from commodity k on node v. It is worth noticing that $\sum_{k' \in K} x_{k',v}$ takes the value of $n(v)$, as it corresponds to the total number of packets that are yet counted by node v.

Model (2.1) has a quadratic objective function. Therefore, optimizing or solving this model may involve dealing with non-linearities, for which specialized algorithms tailored for quadratic optimization problems may be more suitable (see Fathi and Bevrani [2019]). Alternatively, linearization techniques can be employed to facilitate the optimization process.

To linearize the objective function, the traditional way is to introduce a new set of variables $z_{k,k',v} = x_{k',v} \times x_{k,v},\ \forall k, k' \in K, v \in V$ that is linked to the variables $\mathbf{x}$ through the following constraints:

$$x_{k',v} + x_{k,v} \leq 1 + z_{k,k',v}, \quad \forall\, k, k' \in K, v \in V. \tag{2.2}$$

Therefore, we obtain an ILP formulation for the FSAP that reads as follows:

$$\min \quad \sum_{k \in K} \sum_{v \in V} \sum_{k' \in K} \varepsilon_v C_k(v)\ z_{k,k',v} \tag{2.3a}$$

$$x_{k',v} + x_{k,v} \leq 1 + z_{k,k',v} \qquad \forall k, k' \in K, v \in V, \tag{2.3b}$$

$$\sum_{v \in \mathcal{V}(F)} x_{k,v} \geq 1 \qquad \forall k \in K,\ F \in \mathcal{F}_k(\Psi_k), \tag{2.3c}$$

$$x_{k,v} \in \{0,1\} \qquad \forall k \in K,\ v \in V, \tag{2.3d}$$

$$z_{k,k',v} \in \{0,1\} \qquad \forall k, k' \in K,\ v \in V. \tag{2.3e}$$

Flow-Sketch Assignment problem with trade-off between the estimation error and the uncovered paths error

We now present an ILP model for the Flow-Sketch Assignment problem that balances two objectives; minimizing the estimation error and minimizing the error due to uncovered paths.

In the context of OD-pair communication across multiple paths, a strategic approach involves selectively applying sketches to only some paths, rather than all of them. By constraining the packet count on chosen paths, there is a possibility of reducing overlap, leading to a decrease in estimation errors for each sketch. However, the decision to ignore packets on specific paths introduces an additional source of error, called *partial-coverage* error.

Given the binary variables $x_{k,v}$, previously introduced, which are associated with every commodity $k \in K$, on every node $v \in V$, an ILP formulation for the FSAP with a trade-off between the sketch estimation error and the partial-coverage error reads as

follows:

$$\min \sum_{k \in K} \left[\sum_{v \in V} \varepsilon_v x_{k,v} \sum_{k' \in K} x_{k',v} C_k(v) + \sum_{P \in \mathcal{P}(k)} \prod_{v' \in \mathcal{V}(P)} (1 - x_{k,v'}) C^{\pi}(P) \right] \tag{2.4a}$$

$$x_{k,v} \in \{0,1\}, \qquad \forall k \in K,\ v \in V, \tag{2.4b}$$

where $C^{\pi}(P)$ is the path capacity, i.e., $C^{\pi}(P)$ gives the maximum possible load that can cross path P.

In the objective function (2.4a) , the left addendum is the estimation error on deployed sketches, called sketch estimation error or simply sketch error, while the right one is the overestimation due to uncovered paths, called the coverage error. It is worth noticing that this function depends on the node capacity $C_k(v)$, which is an upper bound on the number of packets we may expect to see on node v from flow k. In our model, we do not minimize the actual error, but an upper bound instead. Indeed, the actual error would depend on the number of packets per flow that traverse each node, which is not known in advance. However, if the number of packets that traverse certain nodes is significantly lower than the chosen upper bound, the error could vary considerably and more precisely, the trade-off between the two real errors could behave differently from the trade-off between their upper bounds. In particular, the sketch error and the coverage error behave in an opposite way with regard to the actual number of packets. Specifically, the sketch error increases as the number of packets rises, whereas the coverage error decreases. This is due to the fact that we consistently overestimate the number of packets ($C_k(v)$) on uncovered paths to prevent underestimations. It is worth noticing that an alternative would be to limit the sum of flows on a node according to its capacity. This option will be discussed later. Moreover, one could consider that no packets are routed (flowing) through that path. In this case, the error will be at its maximum. Therefore, the right addendum in (2.4a) can be replaced by its upper bound ($C_k(v)$). In this way, the two errors will both be proportional to the number of packets traversing the node, hence the trade-off between them would be better estimated through a trade-off between their upper bounds.

Let us introduce a new binary variable $y_{k,v}$ associated with every commodity $k \in K$ and every node $v \in V$. This variable is used to identify paths for which a commodity k is not covered by any flow. More precisely, for every commodity $k \in K$, given a path $P \in \mathcal{P}(k)$, if no flow is monitored on any node of the path, then for one node v of P, i.e. $v \in \mathcal{V}(P)$, the variable $y_{k,v}$ takes value 1. Otherwise, $y_{k,v}$ is equal to 0. We model such a relationship as:

$$\sum_{v \in \mathcal{V}(P)} x_{k,v} + y_{k,v} = 1, \quad \forall\, k \in K,\ P \in \mathcal{P}(k). \tag{2.5}$$

Moreover, for every two commodities $k, k' \in K$ and for every node $v \in V$, we consider again the binary variable $z_{k,k',v}$. Therefore, the sketch assignment problem can be modeled using the following formulation:

$$\min \quad \sum_{k\in K}\sum_{v\in V}\left[\varepsilon_v \sum_{k'\in K} z_{k,k',v} + y_{k,v}\right] C_k(v) \tag{2.6a}$$

$$\text{s.t.} \quad \sum_{v\in\mathcal{V}(P)} x_{k,v} + y_{k,v} \geq 1 \qquad \forall\, k \in K, P \in \mathcal{P}(k), \tag{2.6b}$$

$$x_{k,v} + x_{k',v} \leq 1 + z_{k,k',v} \qquad \forall\, k, k' \in K, v \in V, \tag{2.6c}$$

$$z_{k,k',v} \leq x_{k,v} \qquad \forall\, k, k' \in K, v \in V, \tag{2.6d}$$

$$x_{k,v}, y_{k,v} \in \{0,1\} \qquad \forall\, k \in K, v \in V, \tag{2.6e}$$

$$z_{k,k',v} \in \{0,1\} \qquad \forall\, k, k' \in K, v \in V. \tag{2.6f}$$

The objective function (2.6a) minimizes the error. Constraints (2.6b) links variables x to variables y. In other words, these constraints, together with the objective function, ensure that for every commodity $k \in K$, given a path $P \in \mathcal{P}(k)$, if at least one node of P monitors traffic flow, then sketches will be affected to vertices of P according to the placement decided by variables $x_{k,v}$. Otherwise, $y_{k,v}$ is equal to 1 for only one node of P, according to the optimization sense (minimization) of the objective function. Constraints (2.6c) corresponds to Equation 2.5 where the equality has been replaced by ≥ 1, due to the objective sense. Constraints (2.6d) come from the linearization of Model (2.1) containing a quadratic objective function.

It is worth noticing that constraints (2.6b) are in exponential number. Therefore, to solve Model (2.6), one needs to implement a Branch-and-Cut algorithm with an efficient strategy to separate inequalities (2.6b).

Separation problem of inequalities (2.6b) The separation problem for inequalities (2.6b) consists, given a solution $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ with $\mathbf{x}^* \in \mathbb{R}^{|K|\times|V|}$, $\mathbf{y}^* \in \mathbb{R}^{|K|\times|V|}$ and $\mathbf{z}^* \in \mathbb{R}^{|K|\times|K|\times|V|}$, provided by the linear relaxation of Model (2.6), in determining for every commodity $k \in K$ if there exists a path P_k^* between s_k and t_k such that $\sum_{v\in\mathcal{V}(P_k^*)} x^*_{k,v} + y^*_{k,v} < 1$ and, if not, in finding an inequality that is violated by $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$. The identified inequalities are added to the current linear program, which is solved again. This procedure is repeated until no violated inequality is found. The final solution is then optimal for the linear relaxation of the ILP formulation (2.6).

Therefore, for fractional and integer values of x^* and y^*, the separation problem consists in finding a path P_k^* for every commodity k, that minimizes $\sum_{v\in\mathcal{V}(P_k^*)} x^*_{k,v} + y^*_{k,v}$. Hence, separate inequalities (2.6b) can be done in polynomial time by solving a shortest path problem for every commodity.

2.2.4 A bilevel approach for the FSAP

In the previous section, we presented a formulation for which the worst-case traffic is based on the capacity of each node. In particular, all flows on each node are assumed to be equal to the node capacity. However, in reality, this cannot be true since the sum of the flows is bounded by the node capacity. Hence, we propose in this section, a more

realistic model where the sum of the flows in a node is bounded by its capacity. To achieve this, we adopt a bilevel approach aimed at representing a worst-case feasible traffic scenario. This involves integrating our previous model with a maximum flow subproblem.

Therefore, we present the bilevel model associated with the FSAP. The first level permits to determination of the sketch assignment while minimizing the error. The second level aims at maximizing the error while computing a multi-commodity maximum flow. This corresponds to the worst-case routing strategy. In other words, the follower problem aims at finding the routing that generates the biggest error.

There are two sets of decision variables in bilevel models, the first-level decision variables associated with the so-called *leader problem* which affect the second-level decision variables associated with the so-called *follower problem*. For the FSAP, the general idea of the bilevel formulation is to model the worst-case scenario. In this case, the leader problem aims to determine the sketch assignment while minimizing the error. The follower problem aims at maximizing the error while routing the flow for each OD-pairs. This corresponds to the worst case of routing.

In the following, we first linearize the bilevel model by adding new variables. Second, thanks to the primal-dual relationship, we reformulate the problem as a single-level model.

Let us introduce the leader variables:

- $x_{k,v} \in \{0,1\}$ for $k \in K, v \in V$: equals to 1 if the flow k is assigned to a sketch on node v, 0 otherwise.
- $y_{k,v} \in \{0,1\}$ for $k \in K, v \in V$: equals to 1 if the flow k is not assigned on any of the paths that cross node v.

Let $\mathcal{P}(k)$ be a set of paths from s_k to t_k for commodity $k \in K$. Let us introduce the follower variables:

- $\alpha_{P,k}$ for $k \in K, P \in \mathcal{P}(k)$ is a continuous variable representing the amount of flow routed through path P for commodity k
- $C_k(v)$ for $k \in K, v \in V$ is a continuous variable representing the maximum observable load on node v for commodity k

A bilevel formulation for the FSAP reads as follows:

$$\min_{x,\,y} \quad \sum_{k \in K} \sum_{v \in V} \left[\varepsilon_v \sum_{k' \in K} x_{k,v} x_{k',v} + y_{k,v} \right] C_k(v) \tag{2.7a}$$

$$\sum_{v \in \mathcal{V}(p)} x_{k,v} + y_{k,v} \geq 1 \qquad \forall\, k \in K, p \in \mathcal{P}(k) \tag{2.7b}$$

$$x_{k,v}, y_{k,v} \in \{0,1\} \qquad \forall\, k \in K, v \in V \tag{2.7c}$$

$$\max_{C} \quad \sum_{k\in K}\sum_{v\in V}\left[\varepsilon_v \sum_{k'\in K} x_{k,v}x_{k',v} + y_{k,v}\right]C_k(v) \tag{2.7d}$$

$$\sum_{k\in K}\sum_{P\in\mathcal{P}(k):e\in P} \alpha_{P,k} \le c_e \qquad \forall\, e\in E \tag{2.7e}$$

$$\sum_{P\in\mathcal{P}(k):v\in\mathcal{V}(P)} \alpha_{P,k} = C_k(v) \qquad \forall\, v\in V, k\in K \tag{2.7f}$$

$$\sum_{k\in K} x_{k,v}^{(t-1)} C_k(v) \le C(v) \qquad \forall\, v\in V \tag{2.7g}$$

$$\sum_{k\in K} (1 - x_{k,v}^{(t-1)}) C_k(v) \le \bar{C}(v) \qquad \forall\, v\in V \tag{2.7h}$$

$$\alpha_{P,k} \ge 0 \qquad \forall\, k\in K, P\in\mathcal{P}(k) \tag{2.7i}$$

$$C_k(v) \ge 0 \qquad \forall\, k\in K, v\in V \tag{2.7j}$$

where $x_{k,v}^{t-1}$ is a parameter corresponding to the value affected to $x_{k,v}$ at a previous time $(t-1)$, $C(v)$ and $\bar{C}(v)$ are two bounds on the maximum observable load on node v, respectively for assigned and ignored flows. In this bilevel model, the leader problem is given by the first program stated by ((2.7a)-(2.7c)). The objective function (2.7a) aims to minimize the overall error. Constraints (2.7b) of the leader problem link variables x to variables y. Constraints (2.7c) express the definition set of the leader variables x and y. The follower is given by the inner maximization problem stated by ((2.7d)-(2.7j)). The objective function (2.7d) of the follower problem aims to maximize the overall error. Constraints (2.7e) are the capacity constraints imposing that the total flow routed on an edge through paths $P\in\mathcal{P}(k)$, for all commodities $k\in K$, does not exceed the capacity of the edge. Constraints (2.7f) ensure that for every commodity $k\in K$, the load of a node $v\in V$ is equal to the total flow routed through paths $P\in\mathcal{P}(k)$ that contain this node v. Constraints (2.7g) and (2.7h) set limits on the load of a node v, based on the previous sketch assignment (at time $t-1$). Constraints (2.7i) and (2.7j) express the definition set of the follower variables $\boldsymbol{\alpha}$ and $\mathbf{C}$.

We remark that for fixed values of $\mathbf{x}$ and $\mathbf{y}$, the follower problem is a linear program.

To linearize the leader problem, we consider additional variables $z_{k,k',v} = x_{k,v}x_{k',v}$ for all $k,k'\in K$ and $v\in V$. The leader problem can be rewritten as follows:

$$\min \quad \sum_{k\in K}\sum_{v\in V}\left[\varepsilon_v \sum_{k'\in K} z_{k,k',v} + y_{k,v}\right]C_k(v) \tag{2.8a}$$

$$\text{s.t.} \quad \sum_{v\in\mathcal{V}(P)} x_{k,v} + y_{k,v} \ge 1 \qquad \forall\, k\in K, P\in\mathcal{P}(k), \tag{2.8b}$$

$$\sum_{v\in\mathcal{V}(P)} y_{k,v} \le 1 \qquad \forall\, k\in K, P\in\mathcal{P}(k), \tag{2.8c}$$

$$x_{k,v} + x_{k',v} \le 1 + z_{k,k',v} \qquad \forall\, k,k'\in K, v\in V, \tag{2.8d}$$

$$z_{k,k',v} \le x_{k,v} \qquad \forall\, k,k'\in K, v\in V, \tag{2.8e}$$

$$x_{k,v}, y_{k,v} \in \{0,1\} \qquad \forall\, k\in K, v\in V, \tag{2.8f}$$

$$z_{k,k',v} \in \{0,1\} \qquad \forall\, k,k'\in K, v\in V, \tag{2.8g}$$

By introducing the additional variables $\mathbf{z}$, we transform Model (2.7) into a linear bilevel model. However, the main challenge in tackling this model lies in reformulating it into

a single-level model suitable for efficient solvers. This difficulty arises from the min-max relationship, originating from the follower problem, which is a maximization problem while the leader problem involves minimization.

The traditional approach to solving bilevel models, (see Dempe [2020]), involves dualizing the follower problem to determine the objective value to inject into the leader problem. However, this method is unsuitable for our case because we require fixing values of $C_k(v)$ for every commodity $k \in K$ and every node $v \in V$. Therefore, we investigate an alternative approach, also based on the dualization of the follower problem.

Considering four vectors of continuous variables $\boldsymbol{\beta} \in \mathbb{R}^{|E|}$, $\bar{\boldsymbol{\beta}} \in \mathbb{R}^{|V|\times|K|}$, $\boldsymbol{\beta}^1 \in \mathbb{R}^{|V|}$ and $\boldsymbol{\beta}^2 \in \mathbb{R}^{|V|}$, the dual model of the follower problem ((2.7d)-(2.7j)) is given below:

$$\min \quad \sum_{e\in E} c_e\beta_e + \sum_{v\in V} C(v)(\beta_v^1 + \beta_v^2) \tag{2.9a}$$

$$-\bar{\beta}_{v,k} + \sum_{v\in V}(x_{k,v}^{(t-i)})\beta_v^1 + \sum_{v\in V}(1 - x_{k,v}^{(t-i)})\beta_v^2 \geq \varepsilon_v\left[\sum_{k'\in K} z_{k,k',v} + y_{k,v}\right], \qquad \forall v \in V, \forall k \in K, \tag{2.9b}$$

$$\sum_{e\in P}\beta_e + \sum_{v\in\mathcal{V}(P)}\bar{\beta}_{v,k} \geq 0 \qquad \forall k \in K, \forall P \in \mathcal{P}(k), \tag{2.9c}$$

$$\beta_e \geq 0 \qquad \forall e \in E, \tag{2.9d}$$

$$\bar{\beta}_{v,k} \geq 0 \qquad \forall k \in K, \forall v \in V \tag{2.9e}$$

$$\beta_v^1 \geq 0 \qquad \forall v \in V, \tag{2.9f}$$

$$\beta_v^2 \geq 0 \qquad \forall v \in V. \tag{2.9g}$$

The variables $\boldsymbol{\beta}, \bar{\boldsymbol{\beta}}, \boldsymbol{\beta}^1$ and $\boldsymbol{\beta}^2$ are the dual variables associated respectively to Constraints (2.7e), (2.7f), (2.7g), and (2.7h) of the follower problem. Constraints (2.9b) and (2.9c) are associated respectively to variables $\mathbf{C}$ and $\boldsymbol{\alpha}$ of the follower problem.

In what follows, we propose to simultaneously consider both the primal model and dual model of the follower problem by setting their objective value to equality and injecting this model into the leader problem. This results in a reformulation of the follower problem without an objective function, as presented below:

$$\sum_{k\in K}\sum_{v\in V}\left[\varepsilon_v \sum_{k'\in K} z_{k,k',v} + y_{k,v}\right] C_k(v) = \sum_{e\in E} c_e\beta_e + \sum_{v\in V} C(v)(\beta_v^1 + \beta_v^2) \tag{2.10a}$$

$$-\bar{\beta}_{v,k} + \sum_{v\in V}(x_{k,v}^{(t-i)})\beta_v^1 + \sum_{v\in V}(1 - x_{k,v}^{(t-i)})\beta_v^2 \geq \varepsilon_v\left[\sum_{k'\in K} z_{k,k',v} + y_{k,v}\right] \qquad \forall v \in V, \forall k \in K \tag{2.10b}$$

$$\sum_{e\in P}\beta_e + \sum_{v\in\mathcal{V}(P)}\bar{\beta}_{v,k} \geq 0 \qquad \forall k \in K, \forall P \in \mathcal{P}(k) \tag{2.10c}$$

$$\sum_{k\in K}\sum_{P\in\mathcal{P}(k):e\in P}\alpha_{P,k} \leq c_e \qquad \forall\, e \in E \tag{2.10d}$$

$$\sum_{P\in\mathcal{P}(k):v\in\mathcal{V}(P)}\alpha_{P,k} = C_k(v) \qquad \forall\, v \in V, k \in K \tag{2.10e}$$

$$\sum_{k\in K} x_{k,v}^{(t-1)} C_k(v) \leq C(v) \qquad \forall\, v \in V \tag{2.10f}$$

$$\sum_{k\in K}(1 - x_{k,v}^{(t-1)})C_k(v) \leq \bar{C}(v) \qquad \forall\, v \in V \tag{2.10g}$$

$$\alpha_{P,k} \geq 0 \qquad \forall\, k \in K, P \in \mathcal{P}(k) \tag{2.10h}$$

$$C_k(v) \geq 0 \qquad \forall\, k \in K, v \in V \tag{2.10i}$$

$$\beta_e \geq 0 \qquad \forall e \in E, \tag{2.10j}$$

$$\bar{\beta}_{v,k} \geq 0 \qquad \forall k \in K, \forall v \in V \tag{2.10k}$$

$$\beta_v^1 \geq 0 \qquad \forall v \in V, \tag{2.10l}$$

$$\beta_v^2 \geq 0 \qquad \forall v \in V. \tag{2.10m}$$

This model can be added directly to the linearized leader problem ((2.8a)-(2.8g)), which leads to the following single model for the FSAP:

$$\min \quad \sum_{k\in K}\sum_{v\in V}\left[\varepsilon_v \sum_{k'\in K} z_{k,k',v} + y_{k,v}\right] C_k(v) \tag{2.11a}$$

$$\sum_{v\in\mathcal{V}(P)} x_{k,v} + y_{k,v} \geq 1 \qquad \forall\, k \in K, P \in \mathcal{P}(k),$$

$$\sum_{v\in\mathcal{V}(P)} y_{k,v} \leq 1 \qquad \forall\, k \in K, P \in \mathcal{P}(k),$$

$$x_{k,v} + x_{k',v} \leq 1 + z_{k,k',v} \qquad \forall\, k, k' \in K, v \in V,$$

$$-\bar{\beta}_{v,k} + \sum_{v\in V}(x_{k,v}^{(t-i)})\beta_v^1 + \sum_{v\in V}(1 - x_{k,v}^{(t-i)})\beta_v^2 \geq \varepsilon_v\left[\sum_{k'\in K} z_{k,k',v} + y_{k,v}\right] \qquad \forall v \in V, \forall k \in K,$$

$$\sum_{e\in P}\beta_e + \sum_{v\in\mathcal{V}(P)}\bar{\beta}_{v,k} \geq 0 \qquad \forall k \in K, \forall P \in \mathcal{P}(k),$$

$$\sum_{k\in K}\sum_{P\in\mathcal{P}(k):e\in P}\alpha_{P,k} \leq c_e \qquad \forall\, e \in E,$$

$$\sum_{P\in\mathcal{P}(k):v\in\mathcal{V}(P)}\alpha_{P,k} = C_k(v) \qquad \forall\, v \in V, k \in K,$$

$$\sum_{k\in K} x_{k,v}^{(t-1)} C_k(v) \leq C(v) \qquad \forall\, v \in V,$$

$$\sum_{k\in K}(1 - x_{k,v}^{(t-1)})C_k(v) \leq \bar{C}(v) \qquad \forall\, v \in V$$

$$z_{k,k',v} \leq x_{k,v} \qquad \forall k, k' \in K, v \in V,$$

$$x_{k,v}, y_{k,v} \in \{0,1\} \qquad \forall\, k \in K, v \in V,$$

$$z_{k,k',v} \in \{0,1\} \qquad \forall\, k, k' \in K, v \in V,$$

$$\alpha_{P,k} \geq 0 \qquad \forall\, k \in K, P \in \mathcal{P}(k),$$

$$C_k(v) \geq 0 \qquad \forall\, k \in K, v \in V,$$

$$\beta_e \geq 0 \qquad \forall e \in E,$$

$$\bar{\beta}_{v,k} \geq 0 \qquad \forall k \in K, \forall v \in V,$$

$$\beta_v^1 \geq 0 \qquad \forall v \in V,$$

$$\beta_v^2 \geq 0 \qquad \forall v \in V.$$

Discussion

Due to the intricate nature of the bilevel formulation for the FSAP, particularly characterized by a quadratic objective function and nonlinear constraints, deriving a straightforward single-level model is not trivial, and, more specifically, its resolution might be challenging. It is worth noticing that in the literature, some works have been done to address a bilevel optimization problem with non-linear constraints and quadratic objective functions. More precisely, there is a class of bilevel optimization, called *generalized bilevel programming* that allows more flexibility and generality in the modeling of certain problems. Some common features of generalized bilevel programming include non-continuous lower level problem, non differentiable functions and non-convexity, i.e., the upper level and the lower level may be non-convex. We refer the interested reader

to Marcotte and Zhu [1996] and Labbé et al. [1998] for further details on generalized bilevel programming, addressing especially quadratic objective functions and non-linear constraints.

Instead of directly solving the bilevel model or its single-level reformulation, i.e., Model (2.11), we develop a heuristic tailored to address the FSAP, leveraging its inherent bilevel structure.

2.2.5 A greedy algorithm for the FSAP

The general idea of the heuristic proposed to address the FSAP is to use the follower problem from the bilevel model (2.7) to evaluate the solution built from greedy fashion.

In what follows, we give a detailed description of the algorithm, see Algorithm 3.

The algorithm starts with no sketches assigned and a set of uncovered paths for each commodity ($\bar{\mathcal{P}}(k)$). At each iteration, it evaluates the errors introduced by covering a path with a sketch or by not covering a path with a sketch (uncovered paths). These errors, denoted by $\beta_{v,k}$ and $\bar{\beta}_{v,k}$, respectively, are obtained by solving the follower problem ((2.7d)-(2.7j)). More precisely, $\beta_{v,k}$ is the error introduced by covering with a sketch the paths crossing v for commodity k and $\bar{\beta}_{v,k}$ is the error introduced by uncovering the paths crossing v for commodity k. The algorithm then selects the optimal node v^* and commodity k^*, i.e., the node and commodity that minimize $\beta_{v,k}$ and $\bar{\beta}_{v,k}$ and it installs a sketch if the covering is chosen, i.e., if $\beta_{v,k}$ is chosen. In other words, a sketch is assigned on node v to monitor the flow of commodity k when the error introduced by covering the selected path $P \in \bar{\mathcal{P}}(k)$ is less than the error induced by not covering this path. These steps are repeated until all uncovered paths for commodity k have been examined.

Algorithm 3 Flow-Sketch assignment greedy algorithm

procedure INITIALIZE
    No sketch assigned. $\bar{\mathcal{P}}(k) = \mathcal{P}(k), \forall k \in K$.
end procedure
while $\exists k \in K$ such that $\bar{\mathcal{P}}(k) \neq \varnothing$ **do**
    for all $k \in K$ and $v \in V$ covered by $\bar{\mathcal{P}}(k)$ **do**
        Evaluate: $\beta_{v,k}$ and $\bar{\beta}_{v,k}$ by solving the follower problem.
    end for
    Select (v^*, k^*) minimizing β_{v^*,k^*} or $\bar{\beta}_{v^*,k^*}$.
    if β_{v^*,k^*} is selected **then**
        Install sketch on v^* for k^*.
    end if
    Remove paths covered by v^* from $\bar{P}(k^*)$.
end while

2.3 Concluding remarks

In this chapter, we delve into a comprehensive understanding of telecommunication network architecture and the anomalies they may encounter. We then explore the practical applications related to network flow blocker problems, with a primary focus on anomaly detection. Specifically, we demonstrate a direct correlation between the maximum number of anomalies that may occur in the network and the solution of a network flow blocker problem. We then focus on a second application known as network-wide monitoring, which involves observing and analyzing the performance and activity of an entire network infrastructure. By collecting and analyzing data from various network components using sketches, network-wide monitoring enables quick detection of anomalies. Within this context, we investigate the flow-sketch assignment problem, an optimization challenge aimed at strategically placing sketches within a network. To address this problem, we propose a bilevel formulation and a single-level reformulation, followed by the presentation of a greedy algorithm as a solution approach. By solving this problem, network operators aim to achieve efficient resource utilization, ensure reliable connectivity, and deliver high-quality service to end-users.

Chapter 3

The maximum flow blocker problem : Formulations and algorithms

Contents

In this chapter, we study the blocker variant of the maximum problem, namely the maximum flow blocker problem (MFBP). We undertake a comprehensive study of several integer linear programming (ILP) formulations. The first type of model, featuring an exponential number of constraints, is solved through tailored Branch-and-Cut algorithms. In contrast, the second type of ILP model, with a polynomial number of variables and constraints, is solved via a state-of-the-art ILP solver. Using this formulation, we

establish a structural connection between the MFBP and the maximum flow interdiction problem (MFIP). This result enables us to design a novel approach to obtaining solutions for each problem from the other and derive complexity results for both the MFBP and the MFIP.

3.1 The maximum flow blocker problem

In this section, we discuss the MFBP on a directed graph $G = (V, A)$ containing a source $s \in V$ and a destination $t \in V$ and where each arc $a \in A$ is associated with a capacity $c_a \in \mathbb{Z}_+$ and a blocker cost $b_a \in \mathbb{Z}_+$. For this study, as in Section 1.5.2, we focus on graphs where the source is connected to the destination through a single entering arc.

The MFBP consists in finding a minimum-cost subset of arcs to be removed from the graph G, i.e., *blocked*, in such a way that the maximum flow value between s and t in the remaining graph is no larger than a given threshold. This threshold is a positive integer value, called *target-flow* and denoted by $\Phi \in \mathbb{Z}_+$. Without loss of generality, we consider instances in which Φ is less than the maximum flow value in G, i.e, $\Phi < \psi(G)$. Otherwise, clearly, an optimal MFBP solution is the empty subset of blocked arcs.

We introduce two vectors of m positive integer values, the arc-capacity vector $\boldsymbol{c} \in \mathbb{Z}_+^m$ and the blocker cost vector $\boldsymbol{r} \in \mathbb{Z}_+^m$, containing respectively the capacities and blocker costs associated with arcs of the graph G. Accordingly, an instance of the MFBP can be represented as a tuple $(G, \boldsymbol{c}, \boldsymbol{r}, \Phi)$ and we denote by $\zeta(\text{MFBP}(G, \boldsymbol{c}, \boldsymbol{r}, \Phi))$ its optimal solution value.

It is worth noticing that the MFBP is $\mathcal{NP}$-hard and we provide in Section 3.3.4 a reduction from the *maximum flow interdiction problem* (MFIP), which is $\mathcal{NP}$-hard as shown in Wood [1993].

Graphical illustration of MFBP optimal solutions

We illustrate in this section the features of optimal MFBP solutions thanks to the example graph shown in Figure 1.5 with 9 vertices and 16 arcs. As previously, we do not report the arc (t, s), assuming that the values $c_{(t,s)}$ and $b_{(t,s)}$ of its capacity and blocker cost, respectively, are sufficiently large to have no impact on the maximum flow value of the graph nor on the optimal solution value of the MFBP.

We recall that the graph G shown in Figure 1.5 has a maximum flow value of 36 and the minimum cut is given by the set of arcs (v_1, v_5), (v_4, v_6), (v_4, v_7) and (v_3, v_7). It is worth noticing that by removing all the arcs of the minimum cut in a blocker optic, no flow can be routed from the source s to the destination t. In other words, an optimal MCP solution provides a set of arcs of minimum capacity whose removal does not allow any flow from the source to the destination.

The same graph is then used to illustrate an optimal MFBP solution in Figure 3.1. We report on each arc two values separated by the symbol "/": the first one, in blue, is the flow of the arc in the remaining graph; the second one, in black, is the capacity of the arc. Additionally, we report the blocker cost of the arc shown in red. We consider a

target-flow $\Phi = 23$ ($\approx 64\%$ of the maximum flow value in G). In the optimal MFBP solution shown, two arcs are blocked, (v_1, v_5) and (v_2, v_4), with a minimum total blocker cost equal to 6. The blocked arcs are depicted with dashed blue lines. The maximum flow value in the remaining graph is $18 \leq \Phi = 23$. The figure shows with thicker lines the arcs of the minimum cut in the remaining graph. It is worth noticing that not all these arcs and the blocked arcs are contained in the minimum cut of G (see Section 3.3 for further comments on this important point).

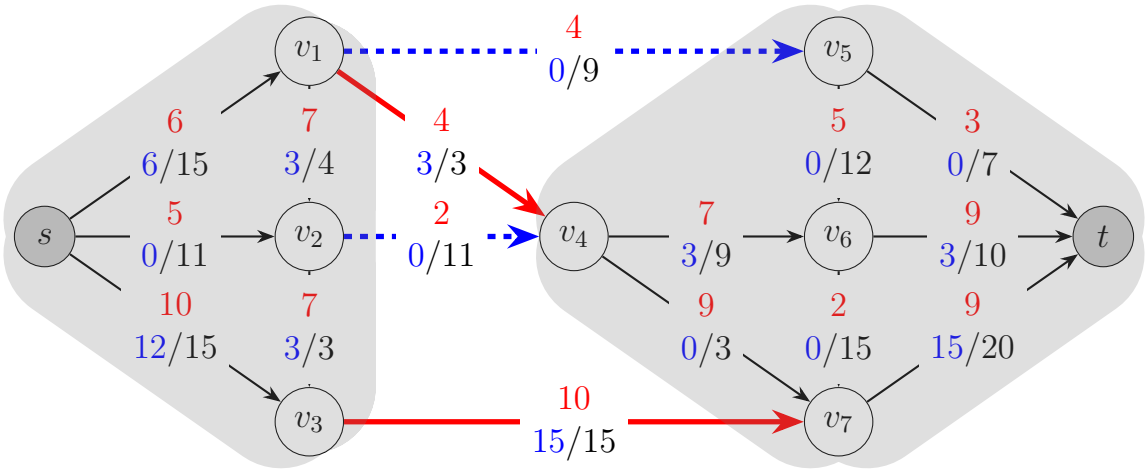


Figure 3.1: An optimal MFBP solution with a target-flow $\Phi = 23$.

3.2 Natural ILP models for the MFBP

In this section we introduce an MFBP model that belongs to the class of blocker models, a special class of bilevel optimization formulations, see e.g., Dempe [2020]. This model and its structural properties are the base of the natural ILP formulations developed in Sections 3.2.2 and 3.2.4, each featuring an exponential family of constraints. These formulations are presented in Bentoumi et al. [2023b] and Bentoumi et al. [2023a]. For each family of constraints, we describe a separation procedure along with its complexity (Sections 3.2.3 and 3.2.5). We also compare the strength of the LP relaxations of the first two formulations (see Section 3.2.6). We then propose in Section 3.2.7 a third ILP formulation featuring the two families of constraints together.

3.2.1 A bilevel formulation

There are two types of variables in blocker models, the first-level decision variables associated with the so-called *leader problem* which affect the second-level decision variables associated with the so-called *follower problem*. For the MFBP, the leader determines a set of blocked arcs to be removed from the graph and the follower determines the maximum flow value in the remaining graph. The leader anticipates the optimal follower's solution to choose the minimum cost subset of arcs to be blocked that results in a remaining graph having a maximum flow value no larger than Φ.

Let us introduce a vector $\boldsymbol{x} \in \{0,1\}^m$ of binary first-level variables m, each variable x_a is associated to an arc $a \in A$ and takes the value 1 if and only if the arc a is blocked, that is, removed from the graph G. A bilevel model for the MFBP reads as follows:

$$\zeta(\text{MFBP}(G, \boldsymbol{c}, \boldsymbol{r}, \Phi)) \;=\; \min_{\boldsymbol{x} \in \{0,1\}^m} \left\{ \sum_{a \in A} r_a \, x_a : \;\; \vartheta(\boldsymbol{x}) \leq \Phi \right\}. \tag{3.1}$$

The link between the leader problem and the follower problem is established by the *value function* $\vartheta(\boldsymbol{x})$ which returns the maximum flow value in the graph containing only the non-blocked arcs. This value is determined by the follower's objective function which also corresponds to the maximum flow value in the graph where the capacity of blocked arcs is set to 0. As far as the leader problem is concerned, the objective function of (3.1) minimizes the total cost of the blocked arcs under the constraint imposing that the maximum flow value $\vartheta(\boldsymbol{x})$ is no larger than Φ.

Let us introduce a second vector $\boldsymbol{y} \in \mathbb{Q}_+^m$ of m non-negative second-level variables, each variable $y_a \geq 0$ is associated to an arc $a \in A$ and it represents the value of the flow on the arc. Therefore, the bilevel model (3.1) for the MFBP can be rewritten as follows:

$$\zeta(\text{MFBP}(G, \boldsymbol{c}, \boldsymbol{r}, \Phi)) \;=\; \min_{\boldsymbol{x} \in \{0,1\}^m} \;\; \sum_{a \in A} r_a \, x_a \tag{3.2a}$$

$$\vartheta(\boldsymbol{x}) \leq \Phi, \tag{3.2b}$$

$$\text{where} \quad \vartheta(\boldsymbol{x}) = \max_{\boldsymbol{y} \in \mathbb{Q}_+^m} \;\; \sum_{a \in \delta^+(s)} y_a \tag{3.2c}$$

$$\sum_{a \in \delta^+(u)} y_a - \sum_{a \in \delta^-(u)} y_a = 0, \quad \forall \, u \in V, \tag{3.2d}$$

$$y_a \leq c_a \, (1 - x_a), \quad \forall \, a \in A. \tag{3.2e}$$

Constraints (3.2d) are the flow conservation constraints, see (1.5b), of the vertices. Constraints (3.2e) model the capacity constraints of the arcs, see (1.5c), and they impose, at the same time, a flow of value 0 on blocked arcs.

A binary realisation $\boldsymbol{x} \in \{0,1\}^m$ of the first-level variables is called a *blocker policy* and it generates a *non-blocked graph* $\mathcal{G}_{NB}(\boldsymbol{x}) = \big(V, \mathcal{A}_{NB}(\boldsymbol{x})\big)$, i.e., the graph with the same vertex set of G and only the non-blocked arcs $a \in A$ with $x_a = 0$ (denoted $\mathcal{A}_{NB}(\boldsymbol{x})$). It is worth noticing that $\vartheta(\boldsymbol{x})$ corresponds to the maximum value of the function $\varphi(\mathcal{G}_{NB}(\boldsymbol{x}))$, i.e., the maximum flow value in the non-blocked graph $\mathcal{G}_{NB}(\boldsymbol{x})$.

In Theorem 2 of Cormican et al. [1998], an important structural property of the follower problem is established. In particular, given a blocker policy $\boldsymbol{x}$, it is proven that the follower problem can be restated as follows:

$$\vartheta(\boldsymbol{x}) = \max_{\boldsymbol{y} \in \mathbb{Q}_+^m} \left\{ \sum_{a \in \delta^+(s)} y_a - \sum_{a \in A} x_a \, y_a : (1.5c), (1.5b). \right\}. \tag{3.3}$$

In this LP reformulation, constraints (3.2e) are replaced with the "standard" capacity constraints (1.5c) for the arcs. The constraints of the follower do not depend anymore on the first-level variables and a penalization term is added to the new objective function to ensure that $\vartheta(\boldsymbol{x})$ is the maximum flow value in the non-blocked graph $\mathcal{G}_{NB}(\boldsymbol{x})$. In other words, Theorem 2 of Cormican et al. [1998] states that the maximum flow value in $\mathcal{G}_{NB}(\boldsymbol{x})$ is equal to the outgoing flow from the source minus the total flow on the blocked arcs. It is worth noticing that the follower problem is an LP formulation featuring totally unimodular system of constraints and accordingly its variables take integer values in its optimal solutions. Moreover, Model (3.3) remains clearly valid also for fractional solution $\boldsymbol{x} \in [0,1]^m$.

3.2.2 A first single-level ILP model for the MFBP

In this section, we introduce the first ILP formulation for the MFBP using only the vector $\boldsymbol{x}$ of binary variables, i.e., the natural variables associated with the arcs whose binary realisations represent the blocker policies. The ILP model is obtained starting from the bilevel model (3.2) and projecting out the variables $\boldsymbol{y}$ of the follower model (3.3). This method is based on a Benders decomposition approach derived from the work of Wood [2011] for general bilevel interdiction problems. In this section, we adapt the approach to address the blocker variant of the MFP. Additionally, we introduce an exact algorithm designed specifically to solve the ILP formulation.

The polytope of feasible solutions for the follower subproblem, which does not depend on the leader variables as shown in (3.3), can be defined as follows:

$$\mathcal{P}_f = \left\{ \boldsymbol{y} \in \mathbb{Q}_+^m : \sum_{a \in \delta^+(s)} y_a \geq \Phi + 1, \quad (1.5c),\ (1.5b). \right\}. \tag{3.4}$$

We remark that only flows of value strictly larger than Φ are associated with the non-dominated constraint (3.2b). This can be imposed by a "$\geq \Phi + 1$" constraint since all the capacities of the arcs are integer values. For this reason, $\mathcal{P}_f$ includes the constraint imposing the lower bound on the value of the flow outgoing from the source. The model (3.3) is valid for any (fractional) vector $\boldsymbol{x} \in [0,1]^m$ and, since the objective function is linear, it is sufficient to optimize over the set of extreme points $\boldsymbol{y}$ of $\mathcal{P}_f$ (denoted $ext(\mathcal{P}_f)$. The constraints (3.2b) can then be restated as follows:

$$\vartheta(\boldsymbol{x}) = \max_{\boldsymbol{y} \in ext(\mathcal{P}_f)} \left\{ \sum_{a \in \delta^+(s)} y_a - \sum_{a \in A} x_a\, y_a \right\} \leq \Phi. \tag{3.5}$$

Accordingly, by applying a Benders-like decomposition to the bilevel model (3.2), we obtain the following single-level ILP reformulation for the MFBP:

$$\zeta(\text{MFBP}(G, \boldsymbol{c}, \boldsymbol{r}, \Phi)) \;=\; \min_{\boldsymbol{x} \in \{0,1\}^m} \quad \sum_{a \in A} r_a\, x_a \tag{3.6a}$$

$$\sum_{a \in \delta^+(s)} y_a - \sum_{a \in A} x_a\, y_a \leq \Phi, \qquad \forall\, \boldsymbol{y} \in ext(\mathcal{P}_f). \tag{3.6b}$$

where constraints (3.2b) are replaced with constraints (3.6b), called *Benders cuts*, an exponential-size family of constraints, one for each extreme point of $\mathcal{P}_f$. This ILP model is called *natural formulation* since it features only the natural variables associated with the arcs and it is denoted by n-ILP$_{\mathtt{B}}$ in the remainder of the article.

To solve n-ILP$_{\mathtt{B}}$, we develop a Branch-and-Benders-Cut approach, i.e., a Branch-and-Cut algorithm where Benders cuts (3.6b) are separated in the nodes of the branching tree for integer and fractional solutions. This exact algorithm requires defining a *relaxed master problem* (RMP) where the binary variables are replaced with continuous variables taking values between 0 and 1. Only a subset of constraints are included in the RMP in the initialization phase. To check that RMP solutions respect all the Benders cuts or to determine one or more violated constraints which are then added to the RMP, we propose the separation procedure described in the next section.

3.2.3 Separation of the Benders cuts

Given a (fractional) solution $\boldsymbol{x} \in [0,1]^m$ of the RMP in a Branch-and-Cut node, the *separation problem* for the Benders cuts (3.6b) requires finding a vector $\boldsymbol{y} \in ext(\mathcal{P}_f)$ such that:

$$\sum_{a \in \delta^+(s)} y_a - \sum_{a \in A} x_a \, y_a > \Phi \tag{3.7}$$

or to prove that such a vector does not exist, i.e., that all Benders cuts are satisfied by the solution $\boldsymbol{x}$. Inequality (3.7) can be equivalently rewritten as follows:

$$\sum_{a \in \delta^+(s)} (x_a - 1) \, y_a + \sum_{a \in A \setminus \delta^+(s)} x_a \, y_a < -\Phi \tag{3.8}$$

and accordingly, it is necessary to find a vector $\boldsymbol{y} \in ext(\mathcal{P}_f)$ leading to the minimum value of the left-hand-side of (3.8). Let us introduce the *minimum cost circulation problem* (MCCP). The MCCP requires a vector $\mathbf{g} \in \mathbb{Q}^m$ where each element g_a is the cost per unit of flow on the arc $a \in A$ and a vector $\mathbf{b} \in \mathbb{Q}^m$ where each element b_a is the lower bound on the flow value passing on the arc $a \in A$. The MCCP aims to find a minimum cost flow respecting the capacity constraints (1.5c), the flow conservation constraints (1.5b), and the lower bounds imposed on the flow of the arcs. By using the vector of continuous variables $\boldsymbol{y} \in \mathbb{Q}_+^m$ for the flow values on the arcs, the MCCP can be modeled by the following LP formulation:

$$\min_{\boldsymbol{y} \in \mathbb{Q}_+^m} \left\{ \sum_{a \in A} g_a \, y_a : (1.5\text{b}), \; b_a \leq y_a \leq c_a, \; \forall \, a \in A \right\}. \tag{3.9}$$

The constraints of this formulation imply the constraints of $\mathcal{P}_f$ and accordingly, the MCCP is used to characterize the complexity of the separation problem in the proof of the following proposition.

Proposition 1. *The separation problem for the Benders cuts* (3.6b) *can be solved in strongly polynomial time for (fractional) solutions* $\boldsymbol{x} \in [0,1]^m$ *of the RMP.*

Proof. We construct a MCCP instance by setting $g_a = (x_a - 1)$ for the arcs $a \in \delta^+(s)$, $g_a = x_a$ for the arcs $a \in A \setminus \delta^+(s)$, $b_{(t,s)} = \Phi + 1$, $b_a = 0$ for every arc $a \in A \setminus \{(t,s)\}$ and the same capacities on the arcs. An optimal solution $\boldsymbol{y}$ of this MCCP instance respects, by construction, the capacity constraints (1.5c), the flow conservation constraints (1.5b) and by setting $b_{(t,s)} = \Phi + 1$, we have $\sum_{a \in \delta^+(s)} y_a \geq \Phi + 1$, i.e., all the constraints of $\mathcal{P}_f$ are satisfied. If the optimal MCCP solution value is strictly smaller than $-\Phi$, then the Benders cut of maximum violation is found and it is associated with $\mathbf{y}$. Otherwise, all Benders cuts are satisfied by solution $\boldsymbol{x}$. The MCCP can be solved in strongly polynomial time, see e.g., Tardos [1985]. Accordingly, the separation problem of the Benders cuts (3.6b) can be solved in strongly polynomial time for any (fractional) solution $\boldsymbol{x}$. □

3.2.4 A second single-level ILP model for the MFBP

In this section, we introduce a second ILP formulation using as previously only the natural variables $\boldsymbol{x}$ associated with the arcs. The ILP model presented is a set-covering type formulation that can be used for solving interdiction and blocker problems. We refer the interested reader to Wei and Walteros [2022] for set-covering formulations of the shortest path interdiction problem, the maximum clique vertex interdiction problem, and the minimum spanning tree interdiction. We then describe the exact algorithm that can be developed to solve this second ILP formulation to proven optimality.

For a given vector $\boldsymbol{y} \in ext(\mathcal{P}_f)$, we define the subset of arcs $\mathcal{A}_S(\boldsymbol{y}) \subseteq A$ routing a strictly positive flow in the extreme point $\boldsymbol{y}$ as follows:

$$\mathcal{A}_S(\boldsymbol{y}) = \Big\{ a \in A : y_a > 0 \Big\}.$$

These arcs induce the *support graph* $\mathcal{G}_S(\boldsymbol{y}) = (V, \mathcal{A}_S(\boldsymbol{y}))$ in which, by construction, the maximum flow value $\psi(\mathcal{G}_S(\boldsymbol{y}))$ is larger than or equal to $\Phi + 1$. We now introduce the following set of constraints called the *target-flow inequalities*:

$$\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1, \qquad \forall\, \boldsymbol{y} \in ext(\mathcal{P}_f). \tag{3.10}$$

For any vector $\boldsymbol{y} \in ext(\mathcal{P}_f)$, the associated constraint (3.10) is valid since it imposes to block at least one arc in the subset $\mathcal{A}_S(\boldsymbol{y}) \subset A$. In other words, the constraint prevents having a flow of value strictly larger than Φ in the support graph $\mathcal{G}_S(\boldsymbol{y})$. Clearly, it is an exponential family of constraints, one for each extreme point $\boldsymbol{y} \in ext(\mathcal{P}_f)$. Accordingly, we obtain the following second natural formulation for the MFBP, denoted by n-ILP$_{\texttt{TF}}$. This formulation reads as follows:

$$\zeta(\text{MFBP}(G, \boldsymbol{c}, \boldsymbol{r}, \Phi)) \;=\; \min_{\boldsymbol{x} \in \{0,1\}^m} \quad \sum_{a \in A} r_a\, x_a \tag{3.11a}$$

$$\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1, \qquad \forall\, \boldsymbol{y} \in ext(\mathcal{P}_f). \tag{3.11b}$$

It is worth noticing that the target-flow inequalities share a similar combinatorial structure with the *super-valid inequalities* (SVI) for bilevel network interdiction models, proposed in Wood [2011]. A technique to lift the right-hand-side (RHS) of the SVI is

described in Wood [2011]. A similar technique can be used to lift the RHS of the target-flow inequalities (3.10), as well. For a given $\boldsymbol{y} \in ext(\mathcal{P}_f)$, in case the following inequality holds:

$$\sum_{a \in \delta^+(s)} y_a - \max_{a,b \in \mathcal{A}_S(\boldsymbol{y})} \{y_a + y_b\} \geq \Phi + 1, \tag{3.12}$$

the RHS of the target-flow inequality can be set to three. This is because blocking the two arcs routing the maximum flow values in $\mathcal{A}_S(\boldsymbol{y})$ is not sufficient to have a maximum flow value $\psi(\mathcal{G}_S(\boldsymbol{y}))$ smaller than or equal to Φ. If inequality (3.12) holds by subtracting only the maximum flow value of the arcs, then the RHS can be set to two. This lifting can be extended to subsets of three or more arcs. However, in our computational experiments (see Chapter 4), only subsets of at most two arcs are found with the desired properties. Since finding the largest two values in $\boldsymbol{y}$ can be done in linear time, the lifting can be performed in a very efficient manner but our extensive preliminary results show that it has a limited beneficial impact on the computational performance.

In Wood [2011], the separation problem of the SVI (which is similar to the separation problem of the target-flow inequalities) is not addressed nor is its computational complexity. For this reason, in the next section, we present the separation problem of the target-flow inequalities (3.10) and characterize its computational complexity. In the computational study presented in Chapter 4, we discuss the impact on the performance of n-ILP$_{\texttt{TF}}$ determined by different exact and heuristic separation procedures based on the separation problem of the target-flow inequalities.

3.2.5 Separation of the target-flow inequalities

Given a (fractional) solution $\boldsymbol{x} \in [0,1]^m$ of the RMP in a Branch-and-Cut node, the *separation problem* for the target-flow inequalities (3.10) requires finding a vector $\boldsymbol{y} \in ext(\mathcal{P}_f)$ such that:

$$\sum_{a \in \mathcal{A}(\boldsymbol{y})} x_a < 1, \tag{3.13}$$

or to prove that such a vector does not exist, i.e., that all target-flow inequalities are satisfied by the solution $\boldsymbol{x}$. Thus, it is necessary to find a vector $\boldsymbol{y} \in ext(\mathcal{P}_f)$ leading to the minimum value of the left-hand-side of (3.13). We distinguish two cases. The first one is for integer RMP solutions and the second one is for fractional solutions.

For an integer solution $\boldsymbol{x} \in \{0,1\}^m$, i.e., for a blocker policy, the separation problem can be restated as an MFP and the next proposition characterizes its computational complexity:

Proposition 2. *The separation problem for the target-flow inequalities* (3.10) *can be performed in strongly polynomial time for integer solutions* $\boldsymbol{x} \in \{0,1\}^m$ *of the RMP.*

Proof. Since $x_a \in \{0,1\}, \forall a \in A$, a violated target-flow inequality can be found if and only if an extreme point $\boldsymbol{y} \in ext(\mathcal{P}_f)$ exists such that the subset of arcs $\mathcal{A}(\boldsymbol{y}) \subset A$ does not contain any blocked arcs, i.e., arcs $a \in A$ for which $x_a = 1$. Accordingly, finding a

violated target-flow inequality can be done by solving the MFP in the non-blocked graph $\mathcal{G}_{NB}(\boldsymbol{x}) = \big(V, \mathcal{A}_{NB}(\boldsymbol{x})\big)$. In case the maximum flow value $\psi(\mathcal{G}_{NB}(\boldsymbol{x}))$ is $\geq \Phi + 1$, an extreme point $\boldsymbol{y} \in ext(\mathcal{P}_f)$ associated to a maximally violated target-flow inequality is found since the left-hand-side of (3.10) is equal to 0. Otherwise, target-flow inequalities are not violated by $\boldsymbol{x}$. The MFP can be solved in strongly polynomial time, see e.g., Ahuja et al. [1993] and Olver and Vegh [2016], and accordingly, the separation problem of the target-flow inequalities (3.10) can be solved in strongly polynomial time for any integer RMP solution $\boldsymbol{x}$.

□

For fractional RMP solutions $\boldsymbol{x}$, let $\boldsymbol{z} \in \{0,1\}^m$ be a vector of binary variables where each variable z_a is equal to 1 if and only if the arc a is in $\mathcal{A}(\boldsymbol{y})$, i.e., if the arc a is used to route a flow in $\boldsymbol{y} \in ext(\mathcal{P}_f)$. The separation problem for the target-flow inequalities can be modeled by the following ILP problem:

$$\min_{\substack{\boldsymbol{z} \in \{0,1\}^m, \\ \boldsymbol{y} \in ext(\mathcal{P}_f)}} \left\{ \sum_{a \in A} x_a \, z_a : \; y_a \leq c_a \, z_a, \; \forall \, a \in A \right\}. \tag{3.14}$$

The objective function minimizes the left-hand side of (3.13). The constraints are the ones of $\mathcal{P}_f$ with the additional constraints to impose to select an arc if it routes a flow in $\boldsymbol{y}$. If the optimal solution value is strictly smaller than 1, a target-flow inequality maximally violated by $\boldsymbol{x}$ is found. Otherwise, no target-flow inequalities are violated by $\boldsymbol{x}$.

Let us introduce the *minimum edge-cost flow problem* (MECFP), which is used to characterize the complexity of the separation problem for fractional RMP solutions. Given an *arc-price* vector $\mathbf{p} \in \mathbb{Z}_+^m$ and a *flow-value bound* $R \in \mathbb{Z}_+$, the MECFP requires finding a minimum-price flow from s to t with a flow value larger than or equal to R. The MECFP is $\mathcal{NP}$-complete in its decision version, see Garey and Johnson [1979], and accordingly, it is $\mathcal{NP}$-hard in its optimization version. It is worth noticing that the MECFP remains $\mathcal{NP}$-hard also when p_a is in $[0, 1]$. By reducing the MECFP to the separation problem, the next proposition characterizes the computational complexity of the latter one.

Proposition 3. *The separation problem for the target-flow inequalities* (3.10) *is $\mathcal{NP}$-hard for fractional solutions $\boldsymbol{x} \in [0,1]^m$ of the RMP.*

Proof. Starting from a MECFP instance, we set $x_a = p_a$ (a value in $[0, 1]$) for every arc $a \in A$ and $\Phi = R - 1$. Once the separation problem (3.14) is solved, its optimal solution $(\boldsymbol{y}, z)$ corresponds to an optimal MECFP solution. Since $\boldsymbol{y}$ is in $ext(\mathcal{P}_f)$, all the MECFP constraints are satisfied, i.e., the flow conservation constraints, the capacity constraints and the requirement of having a flow value larger than or equal to R. The variable values $\boldsymbol{z}$ correspond to the arcs with a minimum-price flow from s to t.

□

Following the idea in proposed in Wood [2011], a violated target-flow inequality (3.10) can be derived from a violated Benders cut (3.6b), which are both associated with a

vector $\boldsymbol{y} \in ext(\mathcal{P}_f)$. More precisely, if $\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a > 1$, a violated target-flow inequality is found and can be added to the RMP. This method for generating violated target-flow inequalities for fractional and integer RMP solutions based on a violated Benders cut is computationally effective, as shown by the tests reported in Chapter 4.

3.2.6 Comparison of the strength of the LP relaxations of the natural formulations

This section compares the strength of the LP relaxations of the two natural formulations n-ILP$_{\mathtt{B}}$ and n-ILP$_{\mathtt{TF}}$, for the MFBP, presented in the previous section. Let us introduce the two polytopes $\mathcal{P}_{n,\mathtt{B}}$ and $\mathcal{P}_{n,\mathtt{TF}}$, defined respectively by the Benders cuts (3.6b) and the target-flow inequalities (3.10), which are formally defined as follows:

$$\mathcal{P}_{n,\mathtt{B}} = \{\boldsymbol{x} \in [0,1]^m : \ \boldsymbol{x} \text{ satisfies (3.6b)}\}, \quad \mathcal{P}_{n,\mathtt{TF}} = \{\boldsymbol{x} \in [0,1]^m : \ \boldsymbol{x} \text{ satisfies (3.10)}\}.$$

The next proposition shows that the upper bounds on $\zeta(\text{MFBP}(G, \boldsymbol{c}, \boldsymbol{r}, \Phi))$ provided by the optimal solution value of the LP relaxations of n-ILP$_{\mathtt{B}}$ and n-ILP$_{\mathtt{TF}}$ do not dominate each other.

Proposition 4. *There are instances of the MFBP for which, $\mathcal{P}_{n,\mathtt{TF}} \subset \mathcal{P}_{n,\mathtt{B}}$, and other ones for which $\mathcal{P}_{n,\mathtt{B}} \subset \mathcal{P}_{n,\mathtt{TF}}$.*

Proof. We consider the graph G represented in Figure 3.2 and we exhibit two instances, one for which the LP relaxation of n-ILP$_{\mathtt{TF}}$ is strictly stronger than the one of n-ILP$_{\mathtt{B}}$ and another for which the opposite holds.

Setting $\Phi = 7$ ($\approx 19\%$ of the maximum flow), the optimal solution value of the LP relaxation of n-ILP$_{\mathtt{TF}}$ is equal to 16 with $x_{(s,v_3)} = x_{(v_1,v_5)} = x_{(v_2,v_4)} = 1$ and $x_a = 0$ for all other arcs, which corresponds to an optimal MFBP solution. For the same instance, the optimal solution value of the LP relaxation of n-ILP$_{\mathtt{B}}$ is equal to $\frac{133}{10}$ with $x_{(v_1,v_5)} = x_{(v_2,v_4)} = 1$, $x_{(v_3,v_7)} = \frac{73}{100}$ and $x_a = 0$ for all other arcs. Therefore, for this instance, we have $\mathcal{P}_{n,\mathtt{TF}} \subset \mathcal{P}_{n,\mathtt{B}}$.

We now consider the graph G with different blocker costs and arc capacities. We associate with the arcs (v_1, v_5) and (v_3, v_7) a capacity equal to 5. We associate with the arcs $(s, v_1), (s, v_2)$ and (s, v_3) a blocker cost equal to 100. All other arcs have unitary blocker costs and capacities. The maximum flow value is now 3. Setting $\Phi = 1$ ($\approx 34\%$ of the maximum flow), the optimal solution value of the LP relaxation of n-ILP$_{\mathtt{TF}}$ is equal to $\frac{3}{2}$, with $x_{(v_5,t)} = x_{(v_6,t)} = x_{(v_7,t)} = \frac{1}{2}$ and $x_a = 0$ for all other arcs, while the optimal solution value of the LP relaxation of n-ILP$_{\mathtt{B}}$ is equal to $\frac{39}{20}$ with $x_{(v_5,t)} = x_{(v_6,t)} = x_{(v_7,t)} = \frac{13}{20}$ and $x_a = 0$ for all other arcs. For this second instance, we have $\mathcal{P}_{n,\mathtt{B}} \subset \mathcal{P}_{n,\mathtt{TF}}$. □

3.2.7 A third single-level ILP model for the MFBP

A third natural formulation for the MFBP can be obtained by using both families of constraints, i.e, the Benders cuts (3.6b) and the target-flow inequalities (3.10). This formulation is denoted by n-ILP$_{\mathtt{B+TF}}$ and it reads as follows:

$$\zeta(\text{MFBP}(G, \boldsymbol{c}, \boldsymbol{r}, \Phi)) \ = \ \min_{\boldsymbol{x} \in \{0,1\}^m} \quad \left\{ \sum_{a \in A} r_a \, x_a : \ \text{(3.6b)}, \ \text{(3.10)} \right\}. \tag{3.15}$$

The performance of all three ILP formulations is empirically evaluated and the results are discussed in the computational section of this manuscript (see Chapter 4). Moreover, the strength of the natural formulations is discussed and compared with the additional formulation introduced in the next paragraph (see Section 3.3.3).

3.3 A compact ILP model for the MFBP

In this section, we first present a compact ILP formulation for the MFBP. This formulation, featuring a polynomial number of variables and constraints, is obtained through the "dualize-and-combine" method (see Wei and Walteros [2022]) applied to the bilevel model introduced in Section 3.2.1, i.e., Model (3.2) and it exploits the nature of the follower problem. We then present a property that establishes a structural link between solutions of the MFBP and solutions of the maximum flow interdiction problem (MFIP). This result allows us to derive some complexity results for the MFBP and the MFIP.

3.3.1 A fourth compact ILP model for the MFBP

By using the vector of binary variables $\boldsymbol{x} \in \{0,1\}^m$ representing the blocked arcs, we derive from Model (3.2), a single-level ILP formulation exploiting the nature of the value function $\vartheta(\boldsymbol{x})$, as defined by Model (3.3), in which the dependency on the $\boldsymbol{x}$ variables is in the objective function of the model.

Given $\boldsymbol{x} \in [0,1]^m$, the objective function of (3.3) can be re-written as $y_{(t,s)} - \sum_{a \in A} x_a \; y_a$. This is due to the fact that the flow on the arc (t,s) is equal to the outgoing flow from the source s. Accordingly, the dual of Model (3.3) reads as follows:

$$\vartheta(\boldsymbol{x}) = \min_{\substack{\boldsymbol{\gamma} \in \mathbb{Q}^n_+, \\ \boldsymbol{\mu} \in \mathbb{Q}^m_+}} \left\{ \sum_{a \in A} c_a \; \mu_a \;\; : \;\; \mu_{uv} + \gamma_v - \gamma_u \geq -x_{uv}, \; \forall \; (u,v) \in A, \; \gamma_s - \gamma_t \geq 1 \right\}. \quad (3.16)$$

This dual LP model features a vector $\boldsymbol{\mu} \in \mathbb{Q}^m_+$ of m dual non-negative variables associated with the capacity constraints (1.5c) and a vector $\boldsymbol{\gamma} \in \mathbb{Q}^n_+$ of n dual non-negative variables associated with the flow conservation constraints (1.5b). The dual Model (3.16) has a totally unimodular system of constraints and accordingly, all variables $\boldsymbol{\mu}$ take binary values in any optimal dual solutions and there exists an optimal dual solution where all variables $\boldsymbol{\gamma}$ take binary values.

By using in (3.2) the value function $\vartheta(\boldsymbol{x})$ as defined in (3.16) and two sets of binary variables $\boldsymbol{\mu} \in \{0,1\}^m$ and $\boldsymbol{\gamma} \in \{0,1\}^n$, we obtain the following bilevel reformulation of Model (3.2):

$$\zeta(\text{MFBP}(G, \boldsymbol{c}, \boldsymbol{r}, \Phi)) \;\; = \;\; \min_{\boldsymbol{x} \in \{0,1\}^m} \;\; \sum_{a \in A} r_a \; x_a \quad (3.17a)$$

$$\vartheta(\boldsymbol{x}) \leq \Phi, \quad (3.17b)$$

$$\text{where} \quad \vartheta(\boldsymbol{x}) = \min_{\substack{\boldsymbol{\gamma} \in \{0,1\}^n, \\ \boldsymbol{\mu} \in \{0,1\}^m}} \quad \sum_{a \in A} c_a \, \mu_a \tag{3.17c}$$

$$\mu_{uv} + \gamma_v - \gamma_u \geq -x_{uv}, \ \forall \, (u,v) \in A \tag{3.17d}$$

$$\gamma_s - \gamma_t \geq 1. \tag{3.17e}$$

It is worth noticing that Constraints (3.17b) impose that the optimal solution value of the follower problem ((3.17c)-(3.17e)) should not exceed the target-flow Φ. We can replace $\vartheta(\boldsymbol{x})$ by the objective function value in (3.17c), i.e., "$\sum_{a \in a} c_a \mu_a$", since by doing so, we impose that there exists at least one solution of the follower problem ((3.17c) - (3.17e)) whose value is smaller than Φ and accordingly, its optimal solution value is also smaller than Φ.

Therefore, we obtain a compact ILP formulation for the MFBP, which reads as follows:

$$\zeta(\text{MFBP}(G, \boldsymbol{c}, \boldsymbol{r}, \Phi)) \ = \ \min_{\substack{\boldsymbol{x}, \boldsymbol{\mu} \in \{0,1\}^m, \\ \boldsymbol{\gamma} \in \{0,1\}^n}} \quad \sum_{a \in A} r_a \, x_a \tag{3.18a}$$

$$\sum_{a \in A} c_a \, \mu_a \leq \Phi, \tag{3.18b}$$

$$\mu_{uv} + x_{uv} + \gamma_v - \gamma_u \geq 0, \qquad \forall \, (u,v) \in A, \tag{3.18c}$$

$$\gamma_s - \gamma_t \geq 1. \tag{3.18d}$$

This ILP formulation (3.18), denoted by c-ILP, is called *compact formulation*, since it features a polynomial number of variables and constraints. Accordingly, it can be directly solved using an ILP solver.

It is worth noticing that in any optimal solution, for a given arc $a \in A$, we can have either $\mu_a = 1$ or $x_a = 1$ but not both. This is due to the fact that $r_a > 0, \forall a \in A$. If $x_a = \mu_a = 1$, then the solution obtained by setting $\mu_a = 0$ and keeping other values unchanged is still feasible and does not increase the objective function value. For this reason and due to the nature of constraints (3.18c) and (3.18d), an optimal solution of (3.18) is a cut in the graph G, denoted by $\delta(U^G(\boldsymbol{x}))$, which depends on an optimal blocker policy $\boldsymbol{x}$. This cut $\delta(U^G(\boldsymbol{x}))$ is given by the arcs $a \in A$ where $\mu_a = 1$ or $x_a = 1$ and it is the union of the set of non-blocked arcs such that $x_a = 0$ and $\mu_a = 1$ and the set of blocked arcs such that $x_a = 1$ and $\mu_a = 0$. If a variable γ_u is equal to 1, it indicates that vertex u is in the subset $U^G(\boldsymbol{x})$ containing the source s and if it is equal to 0, it indicates that vertex u is in the subset $V \setminus U^G(\boldsymbol{x})$ containing the destination t. In addition, any optimal solution $(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\gamma})$ of Model (3.18) contains the minimum cut $\delta(U^{\mathcal{G}_{NB}}(\boldsymbol{x}))$ in the non-blocked graph $\mathcal{G}_{NB}(\boldsymbol{x})$ which is given by the arcs $a \in \mathcal{A}_{NB}(\boldsymbol{x})$ such that $\mu_a = 1$ and $U^{\mathcal{G}_{NB}}(\boldsymbol{x})$ is a set of vertices containing the source s. This is due to the fact that constraints (3.18c) of Formulation (3.18) can be equivalently rewritten as follows:

$$\mu_{uv} + \gamma_v - \gamma_u \geq 0, \ \ \forall \, (u,v) \in \mathcal{A}_{NB}(\boldsymbol{x}), \tag{3.19}$$

where $\mathcal{A}_{NB}(\boldsymbol{x})$ are the arcs of $\mathcal{G}_{NB}(\boldsymbol{x})$. Together with constraints (3.18d), they are the standard MCP constraints for the non-blocked graph $\mathcal{G}_{NB}(\boldsymbol{x})$.

Figure 3.2 illustrates the same optimal MFBP solution as Figure 3.2 with a target-flow $\Phi = 23$. In addition, we report on each arc the values of $\boldsymbol{\mu}$ and $\boldsymbol{x}$ variables associated with an optimal solution $(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\gamma})$ of Model (3.18). The arcs of the minimum cut $\delta(U^{\mathcal{G}_{NB}}(\boldsymbol{x}))$ in the non-blocked graph $\mathcal{G}_{NB}(\boldsymbol{x})$ are depicted with thick red lines, i.e., the arcs $a \in A$ with $\mu_a = 1$. The arcs of $\delta(U^{G}(\boldsymbol{x}))$ are the arcs of $\delta(U^{\mathcal{G}_{NB}}(\boldsymbol{x}))$ together with the blocked arcs depicted with dashed blue lines, i.e., the arcs $a \in A$ with $x_a = 1$.

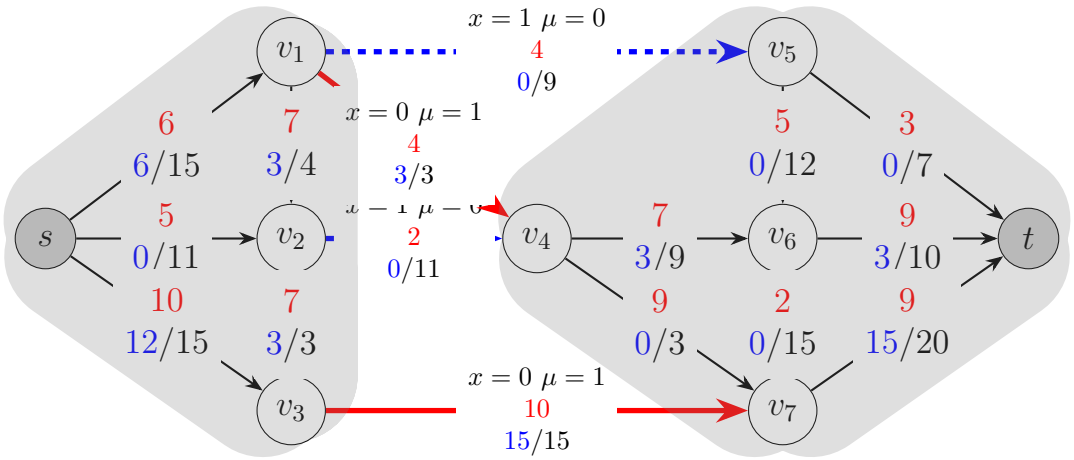


Figure 3.2: The cut $\delta(U^{G}(\boldsymbol{x}))$ (red and blue dashed arcs) associated to an optimal MFBP solution $(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\gamma})$, composed by the blocked arcs (blue dashed arcs) and the minimum cut $\delta(U^{\mathcal{G}_{NB}}(\boldsymbol{x}))$ (red arcs) in the non-blocked graph $\mathcal{G}_{NB}(\boldsymbol{x})$.

Figure 3.2 also shows that the cut $\delta(U^{G}(\boldsymbol{x}))$ containing the optimal MFBP solution does not necessarily correspond to the minimum cut in the graph. Specifically, in the depicted example, the optimal MFBP solution is contained in a cut having a value of 38 and composed by the arcs $(v_1, v_5), (v_1, v_4), (v_2, v_4)$, and (v_3, v_7). In contrast, the minimum cut in the same graph has a value of 36 and it is composed by the arcs $(v_1, v_5), (v_4, v_6), (v_4, v_7)$, and (v_3, v_7), as illustrated in Figure 1.5.

It is worth noticing that the structure of Model (3.18) exhibits a notable similarity to the formulation introduced in Wood [1993] for the maximum flow interdiction problem (MFIP). More precisely, the next section demonstrates the existence of a structural link between the solutions of the MFIP and the solutions of the MFBP.

3.3.2 Solving the MFBP via the MFIP

Before discussing the link between the MFBP and the MFIP, we recall in this section the compact ILP formulation for the MFIP, proposed in Wood [1993] (See Chapter 1).

Given a graph, together with arc-capacities, positive integer *interdiction costs* $q_a \in \mathbb{Z}_+$ on the arcs $a \in A$ and an *interdiction budget* denoted by Ψ, we recall that the MFIP consists in finding a subset of arcs of total interdiction cost no larger than Ψ to be removed from the graph, i.e., *interdicted*, in such a way that the maximum flow value

in the remaining graph is minimized. We introduce the interdiction cost vector $\boldsymbol{q} \in \mathbb{Z}_+^m$ of m positive integer values containing the interdiction costs associated with the arcs of G. Accordingly, an instance of the MFIP can be represented as a tuple $(G, \boldsymbol{c}, \boldsymbol{q}, \Psi)$ and we denote by $\zeta(\text{MFIP}(G, \boldsymbol{c}, \boldsymbol{q}, \Psi))$ its optimal solution value.

Let us introduce $\boldsymbol{w} \in \{0,1\}^m$, which is a vector of binary variables associated with the set of arcs A of G, each variable w_a takes value 1 if and only if the arc a is removed from G, i.e., *interdicted*. A binary realization of variables $\boldsymbol{w} \in \{0,1\}^m$ is called an *interdiction policy* and it generates a *non-interdicted graph* $\mathcal{G}_{NI}(\boldsymbol{w}) = \big(V, \mathcal{A}_{NI}(\boldsymbol{w})\big)$, i.e., the graph induced by the set of non-interdicted arcs $a \in A$ with $w_a = 0$ (denoted $\mathcal{A}_{NI}(\boldsymbol{w})$).

As explained in Chapter 1, in Wood [1993], the author employs a dualize-and-combine approach to derive a compact ILP formulation for the MFIP starting from its bilevel formulation. However, in contrast with the approach detailed in Section 3.2.1, the compact formulation of the MFIP is obtained after computing the dual of the follower problem, which is a MFP where the capacity constraints takes into account the action of the leader. In other words, as for the MFBP, the "standard" capacity constraints (1.5c) of the MFP are replaced by the capacity constraints "$y_a \leq c_a(1 - w_a)$" for all arcs $a \in A$. By fixing values of the interdiction variables, the capacity constraints of the follower problem become independent of the first-level variables, adopting the standard form with a right-hand side of $c_a(1 - w_a)$. This allows to replace the follower problem by its dual formulation, resulting in a single-level reformulation model featuring a quadratic objective function and constraints of the MC. The author then uses linearization techniques, taking advantage of the binary nature of the dual variables.

Using two additional sets of binary variables $\boldsymbol{\beta} \in \{0,1\}^m$ and $\boldsymbol{\alpha} \in \{0,1\}^n$, the compact ILP formulation for the MFIP of Wood [1993] reads as follows:

$$\zeta(\text{MFIP}(G, \boldsymbol{c}, \boldsymbol{q}, \Psi)) = \min_{\substack{\boldsymbol{w},\boldsymbol{\beta} \in \{0,1\}^m, \\ \boldsymbol{\alpha} \in \{0,1\}^n}} \quad \sum_{a \in A} c_a \, \beta_a \tag{3.20a}$$

$$\sum_{a \in A} q_a \, w_a \leq \Psi, \tag{3.20b}$$

$$\beta_{uv} + w_{uv} + \alpha_v - \alpha_u \geq 0, \qquad \forall \, (u,v) \in A, \tag{3.20c}$$

$$\alpha_s - \alpha_t \geq 1. \tag{3.20d}$$

It is worth noticing that constraints (3.20c) and (3.20d) share the same structural form as constraints (3.18c) and (3.18d), respectively. For this reason, an optimal solution of (3.20) is also a cut in the graph G, denoted by $\delta(U^G(\boldsymbol{w}))$, which depends on an optimal interdiction policy $\boldsymbol{w}$ (see also Royset and Wood [2007]). More precisely, the cut $\delta(U^G(\boldsymbol{w}))$ is given by the arcs $a \in A$ where $\beta_a = 1$ or $w_a = 1$ and it is the union of the set of non-interdicted arcs $\mathcal{A}_{NI}(\boldsymbol{w})$ such that $\beta_a = 1$ and $w_a = 0$ and the set of interdicted arcs such that $w_a = 1$ and $\beta_a = 0$. The minimum cut $\delta(U^{\mathcal{G}_{NI}}(\boldsymbol{w}))$ in the non-interdicted graph $\mathcal{G}_{NI}(\boldsymbol{w})$ is given by the arcs $a \in \mathcal{A}_{NI}(\boldsymbol{w})$ such that $\beta_a = 1$. Constraint (3.20b) imposes that the total interdiction cost, i.e., the interdiction cost induced by the set of interdicted arcs, does not exceed the interdiction budget Ψ.

The next proposition shows how to obtain an optimal MFBP solution starting from an optimal MFIP solution, where the interdiction costs are set to the capacities of the

arcs, the capacities are set to the blocker costs and the interdiction budget is set to the target-flow, by solving a MCP in the non-interdicted graph.

Proposition 5. *Given an instance $(G, \boldsymbol{c}, \boldsymbol{r}, \Phi)$ of the MFBP, one of its optimal solution is given by an optimal solution of the MCP in the non-interdicted graph $\mathcal{G}_{NI}(\boldsymbol{w})$ given by an optimal interdiction policy $\boldsymbol{w}$ of the MFIP instance $(G, \boldsymbol{r}, \boldsymbol{c}, \Phi)$.*

Proof. Let $(G, \boldsymbol{c}, \boldsymbol{r}, \Phi)$ and $(G, \boldsymbol{c}, \boldsymbol{q}, \Psi)$ be an instance of the MFBP and the MFIP, respectively. We set $q_a = c_a$ and $c_a = r_a$ for every arc $a \in A$ and $\Psi = \Phi$. Once the MFIP (see Model ((3.20a)-(3.20d))) is solved, its optimal solution $(\boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{\alpha})$ corresponds to an optimal MFBP solution. Indeed, since $\boldsymbol{w} \in \{0,1\}^m, \boldsymbol{\beta} \in \{0,1\}^m$ and $\boldsymbol{\alpha} \in \{0,1\}^n$, the constraints of the MFIP, i.e, constraints (3.20b), (3.20c) and (3.20d) become identical to the constraints of the MFBP, i.e, Constraints (3.18b), (3.18c) and (3.18d), respectively. The variable values $\boldsymbol{\beta}$ corresponds to the blocked arcs, while the variable values $\boldsymbol{w}$ corresponds to the arcs remaining in the cut $\delta(U^G(\boldsymbol{\beta}))$. According to the objective function (3.20a) of the MFIP, the total blocker cost of the blocked arcs will be minimized, leading to the optimal solution value of the MFBP. Moreover, when solving a MFIP, the variable values $\boldsymbol{\beta}$ corresponds to the arcs of the minimum cut in the non-interdicted graph $\mathcal{G}_{NI}(\boldsymbol{w})$, as explained previously. Accordingly, the blocked arcs corresponds to the minimum cut in the non-interdicted graph $\mathcal{G}_{NI}(\boldsymbol{w})$, which ends the proof.

□

The same reasoning can be used to prove, as stated in the next proposition, that an optimal MFIP solution can be found starting from an optimal MFBP solution, where the blocker costs are set to the capacities of the arcs, the capacities are set to the interdiction costs and the target-flow is set to the interdiction budget, by solving a MCP in the non-blocked graph.

Proposition 6. *Given an instance $(G, \boldsymbol{c}, \boldsymbol{q}, \Psi)$ of the MFIP, one of its optimal solution is given by an optimal solution of the MCP in the non-blocked graph $\mathcal{G}_{NB}(\boldsymbol{x})$ given by an optimal blocker policy $\boldsymbol{x}$ of the MFBP instance $(G, \boldsymbol{q}, \boldsymbol{c}, \Psi)$.*

Proof. The proof of this proposition can be readily adapted from the proof developed for Proposition 5. □

Finally, for the optimal solution values of the MFBP and the MFIP, we have the following corollary:

Corollary 1. *Given an instance $(G, \boldsymbol{c}, \boldsymbol{r}, \Phi)$ of the MFBP, its optimal solution value $\zeta(\text{MFBP}(\text{G}, \boldsymbol{c}, \boldsymbol{r}, \Phi))$ is equal to the maximum flow value $\psi(\mathcal{G}_{NI}(\boldsymbol{w}))$ in the non-interdicted graph $\mathcal{G}_{NI}(\boldsymbol{w})$ given by an optimal interdiction policy $\boldsymbol{w}$ of the MFIP instance $(G, \boldsymbol{r}, \boldsymbol{c}, \Phi)$, i.e., we have:*

$$\zeta(\text{MFBP}(\text{G}, \boldsymbol{c}, \boldsymbol{r}, \Phi)) = \psi(\mathcal{G}_{NI}(\boldsymbol{w})). \tag{3.21}$$

Given an instance $(G, \boldsymbol{c}, \boldsymbol{q}, \Psi)$ of the MFIP, its optimal solution value $\zeta(\text{MFIP}(\text{G}, \boldsymbol{c}, \boldsymbol{q}, \Psi))$ is equal to the maximum flow value $\psi(\mathcal{G}_{NB}(\boldsymbol{x}))$ in the non-blocked graph $\mathcal{G}_{NB}(\boldsymbol{x})$ given by an optimal blocker policy $\boldsymbol{x}$ of the MFBP instance $(G, \boldsymbol{q}, \boldsymbol{c}, \Psi)$, i.e., we have:

$$\zeta(\text{MFIP}(\text{G}, \boldsymbol{c}, \boldsymbol{q}, \Psi)) = \psi(\mathcal{G}_{NB}(\boldsymbol{x})). \tag{3.22}$$

In Figure 3.3, we represent an optimal MFIP solution on the same graph shown in Figure 3.2, where the interdiction costs equal the capacities of the arcs and the capacities of the arcs equal the blocker costs. We report on each arc $a \in A$, the interdiction cost in red, the arc capacity in black and the maximum flow value of the non-interdicted graph in blue. For the arcs in the cut $\delta(U^G(\boldsymbol{w}))$ of G associated to an optimal MFIP solution $(\boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{\alpha})$ with an interdiction budget $\Psi = 23$, we also report the values of the $\boldsymbol{w}$ and $\boldsymbol{\beta}$ variables (without reporting the arcs). The arcs of the minimum cut $\delta(U^{\mathcal{G}_{NI}}(\boldsymbol{w}))$ in $\mathcal{G}_{NI}(\boldsymbol{w})$ are depicted with thick red lines, i.e., the arcs (v_1, v_5) and (v_2, v_4). The arcs of $\delta(U^G(\boldsymbol{w}))$ are the arcs of $\delta(U^{\mathcal{G}_{NI}}(\boldsymbol{w}))$ together with the interdicted arcs depicted with dashed blue lines, i.e., the arcs (v_1, v_4) and (v_3, v_7).

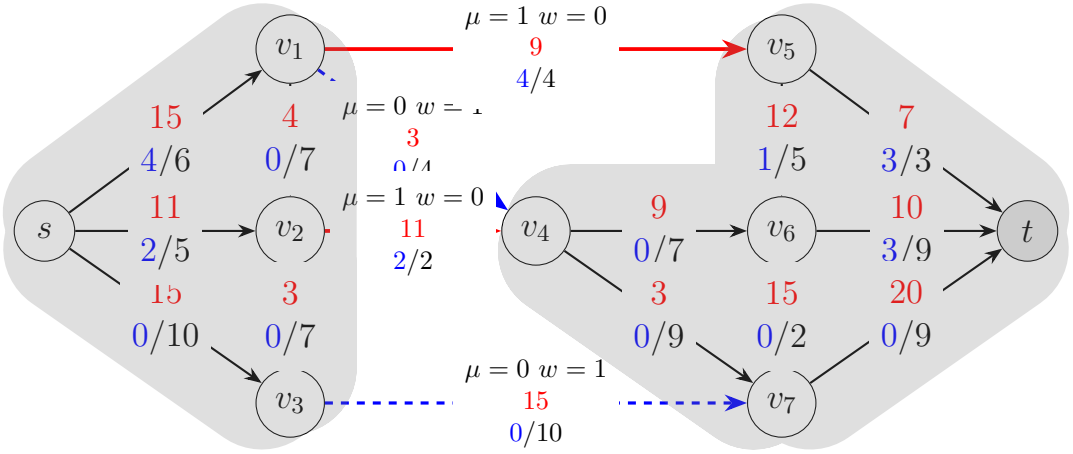


Figure 3.3: The cut $\delta(U^G(\boldsymbol{w}))$ (red and blue dashed arcs) associated to an optimal MFIP solution $(\boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{\alpha})$ with an interdiction budget $\Psi = 23$ and the minimum cut $\delta(U^{\mathcal{G}_{NI}}(\boldsymbol{w}))$ (red arcs) in the non-interdicted graph $\mathcal{G}_{NI}(\boldsymbol{w})$.

Figure 3.3 illustrates Proposition (5). Indeed, the minimum cut $\delta(U^{\mathcal{G}_{NI}}(\boldsymbol{w}))$ corresponds to the set of blocked arcs in the optimal MFBP solution represented in Figure 3.2. In this MFBP example, the target-flow is equal to the interdiction budget, the blocker costs are equal to the arc-capacities and the arc-capacities are equal to the interdiction costs. Moreover, the optimal solution value of the MFBP example represented in Figure 3.2 is equal to 6. As stated by Corollary 1, this value corresponds to the maximum flow remaining in the non-interdicted graph shown in Figure 3.3.

Discussion on the relationship between the MFBP and the MFIP

Proposition 5 and Proposition 6 establish a structural relationship between solutions of the blocker and the interdiction variant of the MFP. More precisely, it proves that solutions of the MFBP can be obtained using any exact algorithm designed for the MFIP, and then solving a MCP in the non-interdicted graph, and vice versa. This is due to the nature of the follower problem that is common for both variants. Indeed, while the blocker problem and the interdiction problem always share the same decision problem, a comparable structural link does not extend to other variants of blocker and interdiction.

For instance, let us consider the blocker and the interdiction variant of the shortest path problem.

Let $G = (V, A)$ be a directed graph where every arc $a \in A$ is associated with a length $l_a \in \mathbb{R}^+$, a delay factor $d_a \in \mathbb{R}^+$ and an interdiction cost $q_a \in \mathbb{R}^+$. Given an interdiction budget Ψ, the shortest path interdiction problem (SPIP) consists in finding a subset of arcs for which the length will be extended by the affected delay, without exceeding the interdiction budget and so that the shortest path between a source $s \in V$ and a destination $t \in V$ is maximized. We refer the reader to Chapter 1 for further details on the shortest path interdiction problem (SPIP). Given an interdiction vector $\boldsymbol{w}$, we recall that a model for the SPIP reads as follows (see Chapter 1):

$$\zeta(\text{SPIP}) = \max_{\boldsymbol{w} \in \{0,1\}^m} \quad \pi_t - \pi_s \tag{3.23a}$$

$$\pi_v - \pi_u - d_a\, w_a \leq 0 \qquad a = (u, v) \in A, \tag{3.23b}$$

$$\pi_s = 0 \tag{3.23c}$$

$$\sum_{a \in A} q_a\, w_a \leq \Psi \tag{3.23d}$$

$$\pi_u \text{ unrestricted} \qquad u \in V \tag{3.23e}$$

Using the same directed graph $G = (V, A)$ where every arc $a \in A$ is associated with a length $l_a \in \mathbb{R}^+$, a delay factor $d_a \in \mathbb{R}^+$ and a blocker cost $r_a \in \mathbb{R}^+$, we consider a *target length* Φ. The shortest path blocker problem (SPBP) consists in finding a subset of arcs for which the length will be extended by the affected delay, so that the shortest path between a source $s \in V$ and a destination $t \in V$ exceeds Φ. The total blocker cost should be minimized.

Given a blocker vector $\mathbf{x} \in \{0, 1\}^m$, we now introduce a bilevel model to solve the shortest path blocker problem (SPBP).

$$\zeta(\text{SPBP}) \;=\; \min \sum_{a\in A} r_a x_a \tag{3.24a}$$

$$\vartheta(x) \geq \Phi \tag{3.24b}$$

$$x_a \in \{0,1\} \qquad a \in A, \tag{3.24c}$$

$$\text{where} \quad \vartheta(w) = \min \sum_{a\in A} (l_a + w_a d_a)\mu_a \tag{3.24d}$$

$$\sum_{a\in\delta^+(u)} x_{(u,v)} - \sum_{a\in\delta^-(u)} \mu_{(u,v)} = \begin{cases} 1 & \text{if } u = s \\ 0 & \text{if } u \in \tilde{V} \quad u \in V, \\ -1 & \text{if } u = t \end{cases} \tag{3.24e}$$

$$\mu_a \geq 0 \qquad a \in A. \tag{3.24f}$$

In this bilevel model, the leader problem is given by the first $0-1$ program stated by ((3.24b)-(3.24c)); and the follower problem is given by the inner minimization program stated by ((3.24d)-(3.24f)). As previously, the leader permits to determine the set of blocked arcs. The follower aims at finding the shortest path in the non-blocked graph, i.e, the graph induced by the set of non-blocked arcs $a \in A$ with $x_a = 0$. The variable $\vartheta(w)$ represents the length of the shortest path in the non-blocked graph. Constraint (3.24b) guarantees that this length is greater than or equal to Φ and constraints (3.24e) are the path constraints.

Using the same approach as for the SPIP, we derive from this bilevel formulation, an MIP for the SPBP that reads as follows:

$$\zeta(\text{SPBP}) \;=\; \max_{\mathbf{x}\in\{0,1\}^m} \sum_{a\in A} r_a x_a \tag{3.25a}$$

$$\pi_v - \pi_u - d_a\, x_a \leq 0 \qquad a = (u,v) \in A, \tag{3.25b}$$

$$\pi_s = 0 \tag{3.25c}$$

$$\pi_t - \pi_s \geq \Psi \tag{3.25d}$$

$$\pi_u \text{ unrestricted} \qquad u \in V \tag{3.25e}$$

It is evident that the structural link established for the MFIP and the MFBP do not apply to the SPIP and the SPBP. Moreover, it is worth noticing that unlike the maximum flow interdiction problem (MFIP) and the maximum flow blocker problem (MFBP), the two models (Model (3.23) and Model (3.25)) for the shortest path interdiction problem (SPIP) and the shortest path blocker problem (SPBP), respectively, do not share the same set of constraints and objective functions. Furthermore, an optimal solution for the SPIP and the SPBP might not be determined based on a specific structure. Hence, establishing a similar relationship between these problems proves challenging.

3.3.3 Comparison of the strength of the LP relaxations of the formulations

This section compares the strength of the LP relaxations of three ILP formulations for the MFBP, presented in the previous sections, namely n-ILP$_{\mathtt{B}}$, n-ILP$_{\mathtt{B+TF}}$ and c-ILP.

Let us consider the polytope $\mathcal{P}_{n,\mathtt{B}}$ presented in Section 3.2.6 and let us introduce the two polytopes $\mathcal{P}_{n,\mathtt{B+TF}}$ and $\mathcal{P}_c$ defined by the constraints of n-ILP$_{\mathtt{B+TF}}$ (3.15) and c-ILP (3.18), respectively, which are formally defined as follows:

$$\mathcal{P}_{n,\mathtt{B+TF}} = \{\boldsymbol{x} \in [0,1]^m \ : \ \boldsymbol{x} \text{ satisfies (3.6b) and (3.10)}\},$$

$$\mathcal{P}_c = \{\boldsymbol{x} \in [0,1]^m, \boldsymbol{\mu} \in [0,1]^m, \boldsymbol{\gamma} \in [0,1]^n \ : \ \boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\gamma} \text{ satisfy (3.18b), (3.18c) and (3.18d)}\}.$$

The next proposition states that the upper bound on $\zeta(\text{MFBP}(G, \boldsymbol{c}, \boldsymbol{r}, \Phi))$ provided by the optimal solution value of the LP relaxation of c-ILP is at least as strong as the upper bound provided by the optimal solution value of the LP relaxation of n-ILP$_{\mathtt{B}}$ and there are instances in which it is strictly stronger. This is due to the fact that the polytope $\mathcal{P}_c$ is strictly contained in the polytope $\mathcal{P}_{n,\mathtt{B}}$ and accordingly, it provides a tighter polyhedral description of the convex hull of the integer solutions.

Proposition 7. *For every instance $(G, \boldsymbol{c}, \boldsymbol{r}, \Phi)$ of the MFBP, we have $\mathcal{P}_c \subseteq \mathcal{P}_{n,\mathtt{B}}$ and this inclusion can be strict.*

Proof. To prove that $\mathcal{P}_c \subseteq \mathcal{P}_{n,\mathtt{B}}$, we consider a point $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\gamma}})$ that belongs to $\mathcal{P}_c$ and we show that $\hat{\boldsymbol{x}}$ is a feasible solution for the natural formulation n-ILP$_{\mathtt{B}}$ (3.6). As explained in the previous section (Section 3.3.1), since $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\gamma}})$ belongs to $\mathcal{P}_c$, it defines a cut in the graph G. This cut is given by the arcs $a \in A$ with $\hat{x}_a = 1$ or $\hat{\mu}_a = 1$. Let $\boldsymbol{y} \in ext(\mathcal{P}_f)$ representing a flow in G. More precisely, for every arc $a \in A$, y_a is the value of the flow on the arc a. Since the value of a flow in a graph is always less than or equal to the capacity of any cut, we have the following inequality:

$$\sum_{a \in A \,:\, \hat{\mu}_a = 1} c_a + \sum_{a \in A \,:\, \hat{x}_a = 1} c_a \geq \sum_{a \in \delta^+(s)} y_a. \tag{3.26}$$

Moreover, as the vector of variables $\hat{\boldsymbol{\mu}}$ satisfies Inequality (3.18b), it follows that $\sum_{a \in A} c_a \, \hat{\mu}_a \leq \Phi$. Accordingly, we have:

$$\Phi + \sum_{a \in A \,:\, \hat{x}_a = 1} c_a \geq \sum_{a \in A \,:\, \hat{\mu}_a = 1} c_a + \sum_{a \in A \,:\, \hat{x}_a = 1} c_a \geq \sum_{a \in \delta^+(s)} y_a,$$

and this can be rewritten as follows:

$$\sum_{a \in \delta^+(s)} y_a - \sum_{a \in A \,:\, \hat{x}_a = 1} c_a \leq \Phi. \tag{3.27}$$

Finally, as for every arc $a \in A$, we have $c_a \geq y_a$, from Equation (3.27), we obtain the following inequality:

$$\sum_{a \in \delta^+(s)} y_a - \sum_{a \in A \,:\, \hat{x}_a = 1} y_a \hat{x}_a \leq \Phi, \tag{3.28}$$

which corresponds exactly to Inequality (3.6b). This shows that $\hat{\boldsymbol{x}}$ is a feasible solution for Model (3.6) and thus $\hat{\boldsymbol{x}}$ is a point of $\mathcal{P}_{n,\mathtt{B}}$.

To prove that the inclusion can be strict, let us go back to the graph example of Figure 3.2. We consider a target-flow $\Phi = 14$. Let $\boldsymbol{x}$ be an optimal solution of the linear relaxation of the natural formulation n-ILP$_{\mathtt{B}}$ (3.6), with $x_{(v_1,v_5)} = \frac{19}{20}$, $x_{(v_2,v_4)} = \frac{81}{100}$, $x_{(v_7,t)} = \frac{1}{5}$ and $x_a = 0$ for all other arcs $a \in A$. The value of this solution is equal to $\frac{361}{50}$. On the other hand, the optimal solution value of the linear relaxation provided by the compact formulation c-ILP (3.18) is equal to $\frac{42}{5} > \frac{361}{50}$. Accordingly, the point $\boldsymbol{x}$ belongs to $\mathcal{P}_{n,\mathtt{B}}$ but does not belong to $\mathcal{P}_c$, which shows that the inclusion can be strict.

□

The next proposition shows that the upper bounds on $\zeta(\text{MFBP}(G, \boldsymbol{c}, \boldsymbol{r}, \Phi))$ provided by the optimal solution value of the LP relaxations of c-ILP and n-ILP$_{\mathtt{B+TF}}$ do not dominate each other.

Proposition 8. *There are instances of the MFBP for which, $\mathcal{P}_c \subset \mathcal{P}_{n,\mathtt{B+TF}}$ and other ones for which $\mathcal{P}_{n,\mathtt{B+TF}} \subset \mathcal{P}_c$.*

Proof. Let us first consider the graph G represented in Figure 3.2. Setting Φ equal to 7 ($\approx 19\%$ of the maximum flow), the optimal solution value of the LP relaxation of the compact formulation c-ILP is equal to $\frac{63}{5}$, with $x_{(v_1,v_5)} = x_{(v_2,v_4)} = 1$, $x_{(v_7,t)} = \frac{66}{5}$ and $x_a = 0$ for all other arcs. For the same instance, the optimal solution value of the LP relaxation of n-ILP$_{\mathtt{B+TF}}$ is equal to 15, with $x_{(v_1,v_5)} = x_{(v_2,v_4)} = x_{(v_7,t)} = 1$ and $x_a = 0$ for all other arcs, which corresponds to an optimal MFBP solution. This proves that for this MFBP instance, we have $\mathcal{P}_{n,\mathtt{B+TF}} \subset \mathcal{P}_c$.

We now consider the same graph G with different values for the arc-capacities and the blocker costs. As previously (see Section 3.2.6), the arcs (v_1, v_5) and (v_3, v_7) have a capacity equal to 5, the arcs $(s, v_1), (s, v_2)$ and (s, v_3) have a blocker cost equal to 100 and all other arcs have unitary blocker costs and capacities. Setting $\Phi = 1$, the optimal solution value of the LP relaxation of n-ILP$_{\mathtt{B+TF}}$ is equal to $\frac{39}{20}$, with $x_{(v_5,t)} = x_{(v_6,t)} = x_{(v_7,t)} = \frac{13}{20}$ and $x_a = 0$ for all other arcs. For the same instance, the optimal solution value of the LP relaxation of c-ILP is equal to 2 with $x_{(v_1,v_5)} = x_{(v_6,t)} = 1$ and $x_a = 0$ for all other arcs, which corresponds to an optimal MFBP solution. This proves that for this second MFBP instance, $\mathcal{P}_c \subset \mathcal{P}_{\mathtt{B+TF}}$.

□

3.3.4 Complexity results for the MFBP and the MFIP

We now investigate the complexity of the MFIP and the MFBP depending on the values of the capacities, the blocker costs, and the interdiction costs on the arcs.

It is worth noting that the MFIP and the MFBP have the same decision problem, which can be expressed as follows: Given a directed graph $G = (V, A)$ with a source $s \in V$ and a destination $t \in V$, where each arc $a \in A$ has a positive integer capacity c_a and a positive integer cost q_a, and two positive integer values Ψ and Φ, there exists a subset of arcs $\mathcal{A} \subseteq A$ such that $\sum_{a \in \mathcal{A}} q_a \leq \Psi$ and whose removal does not allow a flow between s and t of value greater than Φ. Consequently, any complexity result for the MFIP also holds for the MFBP and vice versa.

The next proposition characterizes the computational complexity of the MFBP.

Proposition 9. *The MFBP is strongly $\mathcal{NP}$-hard, even if all arcs have the same blocker cost.*

Proof. In Wood [1993], the author proves that the decision problem of the MFIP is strongly $\mathcal{NP}$-complete even if all arcs have the same interdiction cost. By setting $r_a = q_a$ for all arcs $a \in A$, since the MFIP and the MFBP have the same decision problem, the MFBP is also strongly $\mathcal{NP}$-complete in its decision version and accordingly, it is strongly $\mathcal{NP}$-hard in its optimization version. □

Using Proposition 5, the next proposition provides additional complexity results for the MFIP and the MFBP.

Proposition 10. *The MFIP and the MFBP are strongly $\mathcal{NP}$-hard, even if all arcs have the same capacity.*

Proof. Let $(G, \mathbf{c}, \hat{\mathbf{q}}, \Psi)$ be an MFIP instance, where $\hat{\mathbf{q}} \in \mathbb{Z}_+^m$ is an interdiction cost vector with all values being identical. By Proposition 5, an optimal solution for the MFBP instance $(G, \hat{\mathbf{q}}, \boldsymbol{c}, \Psi)$ can be found in polynomial time by solving a MCP in the non-interdicted graph associated to an optimal solution for the MFIP instance $(G, \boldsymbol{c}, \hat{\mathbf{q}}, \Psi)$. In Wood [1993], the MFIP has been shown to be strongly $\mathcal{NP}$-hard even if all arcs have the same interdiction cost. Accordingly, since the MFBP instance $(G, \hat{\mathbf{q}}, \boldsymbol{c}, \Psi)$ is a MFBP where all arcs have the same capacity, we can deduce that the MFBP is strongly $\mathcal{NP}$-hard, even if all arcs have the same capacity.

Moreover, as the MFIP and the MFBP share the same decision problem, the MFIP is also strongly $\mathcal{NP}$-hard, even if all arcs have the same capacity.

□

3.4 Extensions

This section focuses on variants of the MFBP. Specifically, we are interested in the continuous form of blocker and interdiction problems, which involve the reduction of arc capacities. Additionally, we explore the vertex blocker and interdiction variant, where vertices are removed instead of arcs. Furthermore, we study the blocker problem applied to the maximum cardinality bipartite matching problem, which is called the *maximum cardinality bipartite matching blocker problem*. More precisely, we demonstrate that the equivalence property we previously established between the blocker and interdiction problems remains applicable to these particular variants.

3.4.1 Continuous interdiction and blocker

The previous sections deal with the MFIP and the MFBP, where the master problem aims at removing arcs from the graph. This implies complete destruction of interdicted and blocked arcs. In this case, both the interdiction and blocker problems are said to be

discrete. However, a variant of these problems arises when the master problem aims at reducing the arc-capacities in the graph, resulting in partial destruction of interdicted and blocked arcs. In this context, the interdiction and blocker problems are said to be *continuous*. We denote by MCFBP (resp. MCFIP) the continuous blocker (resp. interdiction) problem applied to the MFP.

Given a directed graph $G = (V, A)$ containing a source $s \in V$ and a destination $t \in V$, a set of capacities $c_a \in \mathbb{Z}_+$, a set of interdiction costs $q_a \in \mathbb{Z}_+$ associated with each arc $a \in A$ and an interdiction budget Ψ, the MCFIP consists in decreasing capacities of arcs such that the maximum flow in the non-interdicted graph is minimum. To solve this problem, we associate with each arc $a \in A$, a variable $\bar{w}_a \in [0, 1]$ corresponding to the percentage of destruction of the arc. If $\bar{w}_a$ is equal to 0, then the capacity of arc a remains the same. If $\bar{w}_a$ is equal to 1, then the arc has been removed. Otherwise, capacity and interdiction cost of arc a is equal respectively to $c_a \times (1 - \bar{w}_a)$ and $b_a \times (1 - \bar{w}_a)$. It is worth noticing that the interdiction cost b_a is that of completely destructing arc a. Moreover, the non-interdicted graph, denoted by $\mathcal{G}_{NI}(\bar{\boldsymbol{w}})$, is identical to the graph G; that is, $\mathcal{G}_{NI}(\bar{\boldsymbol{w}})$ and G share the same set of vertices and arcs, yet with new lower arc capacities.

To address the MCFIP, we can employ the identical bilevel model as used for the MFIP, specifically Model (1.10). The follower problem remains unchanged. The only distinction arises from the master problem's variables $\bar{\boldsymbol{w}}$, which now take values in the set $[0, 1]^m$. Using the same notations as previously, a bilevel model for the MCFIP reads as follows:

$$\zeta(\text{MCFIP}) = \min_{\bar{\boldsymbol{w}} \in [0,1]^m} \ \vartheta(\bar{w}) \tag{3.29a}$$

$$\sum_{a \in A} q_a \, \bar{w}_a \leq \Psi \tag{3.29b}$$

$$\text{where} \quad \vartheta(\bar{w}) = \max_{\mathbf{y} \, \in \, \mathbb{Q}_+^m} \quad \sum_{a \, \in \, \delta^+(s)} y_a \tag{3.29c}$$

$$\sum_{a \, \in \, \delta^+(u)} y_a - \sum_{a \, \in \, \delta^-(u)} y_a = 0, \qquad \forall \, u \in V, \tag{3.29d}$$

$$y_a \leq c_a \, (1 - \bar{w}_a), \qquad \forall \, a \in A. \tag{3.29e}$$

$$y_a \geq 0, \qquad \forall \, a \in A. \tag{3.29f}$$

Applying the same technique as the one employed for the MFIP, we can reformulate Model (3.29) as a compact single-level formulation. This leads to the following model for the continuous interdiction.

$$\zeta(\text{MCFIP}) \;=\; \min \quad \sum_{a\in A} c_a\,\beta_a \tag{3.30a}$$

$$\bar{\beta}_{uv} + \bar{w}_{uv} + \alpha_v - \alpha_u \geq 0 \qquad (u,v)\in A, \tag{3.30b}$$

$$\alpha_s - \alpha_t \geq 1 \tag{3.30c}$$

$$\sum_{a\in A} q_a\,\bar{w}_a \leq \Psi \tag{3.30d}$$

$$\alpha_u \in \{0,1\} \qquad u\in V, \tag{3.30e}$$

$$\bar{\beta}_a \in [0,1] \qquad a\in A, \tag{3.30f}$$

$$\bar{w}_a \in [0,1] \qquad a\in A. \tag{3.30g}$$

This model is similar to Model (3.20). The difference lies in the definition set of variables $\boldsymbol{\beta}$ and $\bar{\boldsymbol{\beta}}$, as well as $\boldsymbol{w}$ and $\bar{\boldsymbol{w}}$.

It is worth noticing that for an arc $a \in A$, variable α_a can take value 0 or 1. Thus, we can have either $\bar{\beta}_a + \bar{w}_a = 0$ or $\bar{\beta}_a + \bar{w}_a = 1$. Therefore, as for the MFIP, any feasible solution can be interpreted as a cut $\delta(U^G(\bar{\boldsymbol{w}}))$ in the graph G, given by arc a where $\bar{\beta}_a + \bar{w}_a = 1$. Consequently, Theorem 5 is still valid for the continuous version of the blocker and the interdiction problems applied to the MFP.

For every arc $a \in A$, let $\bar{x}_a$ be a continuous variable taking values between 0 and 1. Using the same notations as the one employed in Section 3.3, a model for the MCFBP reads as follows:

$$\zeta(\text{MCFBP}) \;=\; \min_{\bar{\mathbf{x}},\bar{\boldsymbol{\mu}}\in\{0,1\}^m,\boldsymbol{\gamma}\in\{0,1\}^n} \quad \sum_{a\in A} b_a\,\bar{x}_a \tag{3.31a}$$

$$\bar{x}_{uv} + \bar{\mu}_{uv} + \gamma_v - \gamma_u \geq 0, \qquad \forall\,(u,v)\in A, \tag{3.31b}$$

$$\gamma_s - \gamma_t \geq 1, \tag{3.31c}$$

$$\sum_{a\in A} c_a\,\bar{\mu}_a \leq \Phi. \tag{3.31d}$$

As for the MCFIP, this model is similar to Model (3.18). The difference lies in the definition set of variables $\boldsymbol{\mu}$ and $\bar{\boldsymbol{\mu}}$, as well as $\mathbf{x}$ and $\bar{\mathbf{x}}$.

In Burch et al. [2006], the authors study the MCFIP under the name *Network Inhibition Problem* and show that the problem is strongly $\mathcal{NP}$-Hard, which leads to the following result.

Proposition 11. *The MCFBP are strongly $\mathcal{NP}$-Hard.*

Proof. As for the discrete interdiction and blocker problem, the MCFBP and the MCFIP have the same decision problems. Since the MCFIP has been shown to be $\mathcal{NP}$-Hard, the MCFBP is also $\mathcal{NP}$-Hard. □

3.4.2 Vertex interdiction and blocker problems

In this section, we discuss the vertex variant of the blocker problem and the interdiction problem, called *maximum flow vertex blocker problem* (V-MFBP) and *maximum flow vertex interdiction problem* (V-MFIP), respectively. In this variant, the master problem seeks to destruct vertices instead of arcs.

A closely related problem, called the *n Most Vital Vertices Problem* is addressed in Corley and Chang [1974]. This problem asks to find a subset of exactly n vertices to be interdicted in order to minimize the maximum flow. This problem is similar to the V-MFIP except that it features a single cardinality constraint instead of a budget constraint and unitary interdiction costs. The variant of finding the n most vital arcs is studied in Ratliff et al. [1975]. In Corley and Chang [1974], the authors show that finding the n most vital arcs is equivalent to finding the n most vital vertices. More specifically, they propose a method to transform a graph G into an augmented graph G^* such that any solution of the n most vital vertices in G corresponds to a solution of the n most vital arcs in G^*.

Given a directed graph $G = (V, A)$, we introduce the augmented graph $G^* = (V^*, A^*)$ constructed from $G = (V, A)$ as follows. For each vertex $v \in V$, we define one vertex $v' \in V^*$ such that $(v, v') \in A^*$ with $q_{(v,v')} = q_v$ and $c_{(v,v')} = max\{c_{(u,v)} : (u, v) \in A\}$. Each arc $(v, u) \in A$ is replaced in the augmented graph by an arc $(v", u) \in A^*$ with $q_{(v",u)} = M$ and $c_{(v",u)} = c_{(v,u)}$, where M is an integer value that is strictly greater than Ψ, the interdiction budget.

The next proposition shows that this transformation also holds when affecting costs on the vertices of the graph.

Proposition 12. *Solving the V-MFIP in a graph G leads to solve a MFIP in the augmented graph G^*.*

Proof. We now show that solving a V-MFIP in G is equivalent to solve a MFIP in G^*. Solving a MFIP in G^* leads to find a minimum subset of arcs to remove from G^* in such a way that the total interdiction cost is less than or equal to the interdiction budget. It is worth noticing that only arcs $(v, v') \in V^*$ with $q_{(v,v')} < M$ can be interdicted. Otherwise, the interdiction budget is exceeded. However, these arcs are represented by a set of vertices in G. Moreover, the maximum flow computed in G^* is equal to the maximum flow value computed in G. Therefore, any feasible solution for the V-MFIP solved in G can be found by solving a MFIP in G^*.

□

This proposition leads to the following corollary regarding the MFBP and the V-MFBP.

Corollary 2. *Solving the V-MFBP in a graph G leads to solve a MFBP in the augmented graph G^*.*

From Proposition (12) and Corollary (2), we can extend Theorem (5) for the continuous variant of interdiction and blocker. Which leads to Theorem (1).

Theorem 1. *Given an optimal solution* $(\boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{\alpha})$ *of Model* (3.20) *for the MFIP instance* $(G^*, \mathbf{b}, \mathbf{c}, \Phi)$*, an optimal solution for the V-MFBP instance* $(G, \mathbf{c}, \mathbf{b}, \Phi)$ *is given by the vertices* $v \in V$ *with* $\beta_{(v,v')} = 1$ *for every arc* $(v, v') \in A^*$*.*

Finally, the following corollary derives directly from Proposition (12) and Corollary (2).

Corollary 3. *The V-MFIP and the V-MFBP are strongly NP-Hard.*

3.4.3 The maximum cardinality bipartite matching blocker problem

Given an undirected graph $G = (V, E)$, a matching is a set of edges with no common vertices. The *maximum cardinality matching problem* aims at finding a matching with maximum cardinality.

The interdiction version of this problem, called the *matching interdiction problem* (MIP), has been studied in Zenklusen [2010]. The author shows that this problem is NP-Complete even when all arcs have the same weight or interdiction cost. He proposes a pseudo-polynomial algorithm to solve the MIP. The MIP finds a set of arcs to be removed from a graph, with respect to the interdiction budget, in order to minimize the cardinality of the maximum matching. The blocker version of this problem consists in finding a minimum set of arcs to be destructed in such a way that the number of arcs in the maximum matching is no larger than a given value.

Let $G = (V_1 \cup V_2, E)$ be a bipartite graph, *i.e*, a graph whose vertices can be divided into two independents sets V_1 and V_2 such that every edge $e \in E$ connects one vertex in V_1 to one vertex in V_2. A particular case of the maximum cardinality matching problem consists in considering as input a bipartite graph. This problem is called maximum cardinality bipartite matching problem (MCBMP). A well-known result shows that the MCBMP can be reduced to a MFP using a simple graph transformation, see Ahuja et al. [1993]. Hence, solving a maximum cardinality bipartite matching interdiction or blocker problem leads to solving a MFIP or a MFBP. Therefore, as for the interdiction and blocker problems applied to the MFP, there exists an equivalence between interdiction and blocker problems applied to the MCBMP. This equivalence is given by Theorem 5.

Finally, the maximum cardinality bipartite matching interdiction problem and the maximum cardinality bipartite matching blocker problem can be formulated as Model (3.20) and Model (3.18), respectively.

3.5 Concluding remarks

In this chapter, we studied the maximum flow blocker problem. Utilizing the natural bilevel formulation for the problem, we introduced four linear integer programming formulations designed for the first time to address the MFBP. The first three formulations, involving only the natural variables associated with the arcs, are solved using the exact algorithms described in the paper. The fourth and last formulation is a compact version solved using a state-of-the-art ILP solver. The natural (non-compact) formulations feature two exponential families of constraints. The first one, with separation being polynomial for fractional and integer solutions, is referred to as Benders cuts. The second one, with separation being $\mathcal{NP}$-hard for fractional solutions, is referred to as target-flow inequalities. Using these results, we developed a tailored Branch-and-Cut algorithm enhanced with Benders cuts and target-flow inequalities. Additionally, using the compact formulation, we established the first connection between the maximum flow blocker problem and the maximum flow interdiction problem. In particular, we have shown that solutions of one problem can be found using solutions of the other. This property has been extended to several variants of the problem, in particular the continuous blocker variant and the vertex blocker variant. Furthermore, we conducted a study to compare the strength of the LP relaxations of the proposed ILP formulations.

In the next chapter, we present a comprehensive computational analysis aimed at comparing performance of the formulations proposed.

Chapter 4

The maximum flow blocker problem : Computational experiments

Contents

In the previous chapter, we presented and theoretically compared several formulations to address the maximum flow blocker problem (MFBP). We first designed three natural formulations, namely n-ILP$_{\mathtt{B}}$ (see Model (3.6)), n-ILP$_{\mathtt{TF}}$ (see Model (3.11)) and n-ILP$_{\mathtt{B+TF}}$ (see Model (3.15)). These formulations feature an exponential number of constraints and they are solved through tailored Branch-and-Cut algorithms. We then presented another formulation having a polynomial number of variables and constraints. This formulation, referred to as c-ILP (see Model (3.18)), is solved via a state-of-the-art ILP solver. In this chapter, the exact methods proposed for the MFBP are compared thanks to an extensive computational campaign on a set of real-world and synthetic instances. Our tests aim at evaluating the performance of the exact algorithms and at determining the features of the instances which can be solved to proven optimality.

4.1 Computational experiments

In this chapter, we present the results of our computational campaign for solving the MFBP. The aim is to assess the performance of the ILP formulations for the MFBP presented in this article, i.e., the natural formulations n-ILP$_{\texttt{B}}$ (see Model (3.6)), n-ILP$_{\texttt{TF}}$ (see Model (3.11)), n-ILP$_{\texttt{B+TF}}$ (see Model (3.15)) and the compact formulation c-ILP (see Model (3.18)). The first three models feature an exponential number of constraints. Accordingly, they are solved via a Branch-and-Cut algorithm. The last model has a polynomial number of variables and constraints. It is solved directly using an ILP solver. Moreover, since there are no exact algorithms existing in the literature for addressing the blocker variant of the MFP, the current state-of-the-art approach involves employing an online-accessible solver designed for bilevel programs. Therefore, we compare this technique to the ILP formulations proposed in the paper.

The specific goals of our computational campaign are threefold. The first is to assess the best configuration of the Branch-and-Cut algorithms for the natural formulations n-ILP$_{\texttt{B}}$ and n-ILP$_{\texttt{TF}}$ and to determine the potential effectiveness of enhancing n-ILP$_{\texttt{B}}$ with a second set of constraints, namely the target-flow inequalities (3.10) (see Model n-ILP$_{\texttt{B+TF}}$ (3.15)). The second involves conducting performance comparisons between the best Branch-and-Cut algorithm for the natural formulations and the direct use of an ILP solver on the compact formulation c-ILP. In addition, we computationally analyze the quality of the MFBP lower bounds provided by the Linear Programming (LP) relaxations of n-ILP$_{\texttt{B}}$, n-ILP$_{\texttt{TF}}$, n-ILP$_{\texttt{B+TF}}$ and c-ILP. The third and final goal is to determine the maximum size of MFBP instances that can be solved to proven optimality using the exact methods presented in this paper. Specifically, we investigate the practical computational difficulty of the main features of MFBP instances.

4.1.1 Implementation's features

The experiments are conducted on a processor Intel Core i5-3340M CPU of 2.70GHz $\times$ 4. To tackle the two ILP formulations, we use `CPLEX12.6.3.0` (called `CPLEX` for brevity) in `C++`, one of the state-of-the-art commercial solvers for ILP problems. The Branch-and-Cut for n-ILP is implemented using the `CONCERT TECHNOLOGY` of `CPLEX` and c-ILP is directly solved using this solver. In our experiments, all computations are performed in a single-thread mode with default values for all `CPLEX` parameters. We employ the open-source `C++` library, called `LEMON` for the implementation of graph data structures. This library offers a comprehensive set of efficient components specifically designed for graphs and network algorithms. For example, we use `LEMON` to solve maximum flow problems.

4.1.2 Benchmark set of MFBP instances

To the best of our knowledge, this is the first article that presents exact methods to solve the MFBP and no instances can be found in the literature. We have created synthetic MFBP instances and produced some starting from instances of a related problem, i.e.,

the MFIP, that have been proposed in Royset and Wood [2007]. The new MFBP instances are characterized by two main components: *i*) the *graph-structural* features and *ii*) the *flow-blocker* features. Creating a well-diversified and representative set of MFBP instances is a challenging task. We explain in what follows the specific choices we adopted in creating our benchmark set of MFBP instances. The graph-structural features of MFBP instances concern the characteristics of the input directed graph $G = (V, A)$ on which the MFBP is addressed. The two principal ones are the number n of vertices in V and the number m of arcs in A. The arc set A is also characterized by an arc-density $d(G) = |A|/(n^2 - n)$. In addition, it is necessary to determine a source node $s \in V$ and a destination node $t \in V$. MFBP instances, as well as MFIP instances, have capacities on the arcs. The distribution of the arc capacities determines the value $\psi(G)$ of the maximum flow between the source s and the destination t. The flow-blocker features of MFBP instances consist in determining the blocker cost $b_a \in \mathbb{Z}_+$ for each arc $a \in A$ and the target-flow $\Phi \in \mathbb{Z}_+$. We have chosen to determine the target-flow by setting it as a percentage $\lambda \in [0, 1]$ of the maximum flow value $\psi(G)$ in the given instances. By opting for this approach, we can conduct an analysis of the computational complexity of MFBP instances based on the maximum flow value allowed in the non-blocked graph, which is determined as a percentage of the maximum flow value in the original graph G. It is worth noticing that if $\lambda = 0$, the value of the target-flow Φ is equal to 0, which implies that no flow between s and t should remain in the non-blocked graph. In this case, the MFBP can be reduced to a MCP. On the other hand, if $\lambda = 1$, the value of the target-flow Φ is equal to the maximum flow value in G, i.e., $\Phi = \psi(G)$, implying that there is no need to remove any arcs from G. The distribution of the blocker costs on the arcs determines the value of the optimal solution for the MFBP.

As far as the graph-structural features of MFBP instances are concerned, we consider three classes of instances: *i*) `GRID` graphs, *ii*) `REAL-NETWORKS` graphs, *iii*) `SYNTHETIC` graphs.
The first two classes are the instances used in the computational study conducted in Royset and Wood [2007]. The third class was generated from a random graph generator that we implemented in `Python 3.8.10`.

1. The `GRID` class consists in rectangular grid networks. In other words, the set of vertices are connected by the set of arcs in the form of a grid of n_1 rows and n_2 columns such that $n = n_1 n_2 + 2$ and $|A| = 2n_1 + 2((n_2 - 1)n_1 + (n_1 - 1)n_2)$. The source s and the destination t are on both sides of the grid. Figure 4.1 illustrates an instance from this class. Infinite capacities are assigned to the arcs connecting the source or the destination. All the other arcs have a capacity drawn randomly from the discrete uniform distribution on $[1, 49]$. In the same manner, the arcs outgoing the source or entering the destination cannot be interdicted. For all other arcs, the interdiction costs are generated by drawing a number from the discrete uniform distribution on $[1, 3]$. This class contains a total of 85 graphs.

   This class contains a total of 17 graphs.

2. The `REAL-NETWORKS` class consists of networks representing highways and roads of two cities in the northeastern United States. In these instances, there are

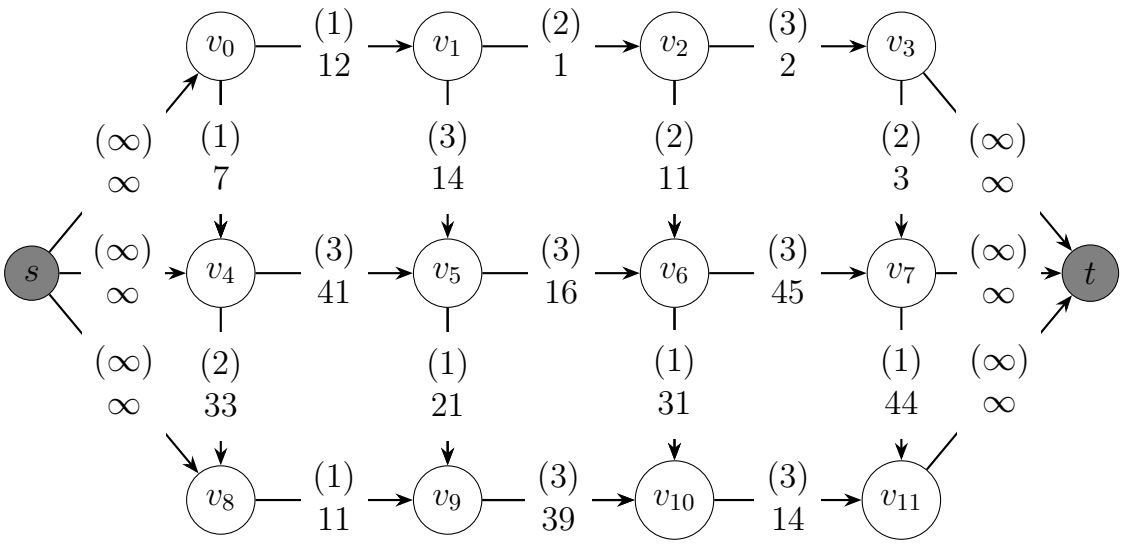


Figure 4.1: Example of a grid network with $n_1 = 3$ and $n_2 = 4$

multiple source vertices and destination vertices, connected respectively to a super-source s and a super-destination t with infinite blocker costs and infinite capacities. This class is categorized into two types, denoted as `Type A` and `Type B` featuring the same graph structure of 3670 vertices and 9876 arcs but different source and destination vertices. Each type consists of seven instances, with distinct values assigned to the blocker costs and capacities on the arcs. For every arc that is not connected to the super-source or to the super-destination, the blocker cost can take three possible values; 1, 2, or 3 and the capacity is drawn randomly from the discrete uniform distribution $[1, 49]$. This class contains a total of 14 graphs.

3. The `SYNTHETIC` class consists of Erdős-Rényi random graphs denoted as $G(n, p)$, where the parameter p determines the probability of including each arc. In this case, the value of p is set to 0.5, meaning that each arc has a 50% chance of being included in the graph. We consider a large set of instance sizes based on the number of vertices n in the graph ranging from 20 to 2000. Specifically, n takes the following values: 20, 30, 40, 50, 60, 100, 300, 500, 700, 900, 1000, 1100, 1200, 1300, 1400 and 1500. The size of the instances also varies depending on the arc densities $d(G)$, which range from 0.1 to 0.9 with a step size of 0.1. For each combination of n and $d(G)$, we create a total of 5 distinct Erdős-Rényi graphs using different random seed generators. The source s and the destination t are randomly selected between the vertices of the graphs such that s and t are linked by at least three consecutive arcs. As for the two previous classes, the arcs outgoing s or entering t have infinite capacities and blocker costs. All the other arcs have a capacity value (resp. blocker cost value) generated by drawing a number from the range [1, 49] (resp. [1, 3]) uniformly at random and rounding it to the nearest integer value. This class contains a total of 420 graphs.

It is worth noticing that, as stated at the beginning of the paper, all instances have an arc (t, s) connecting the destination t to the source s. This arc is given a large

capacity, which can be established by summing the capacities of the arcs outgoing the source. Accordingly, this arc does not impact the value of the maximum flow in the graph. Moreover, it has an infinite blocker cost, ensuring that it cannot be blocked. Consequently, the arc (t, s) does not affect the solution of the MFBP.

In the remainder of this paper, all computing times are expressed in seconds. We impose a CPU time limit of 600 seconds for every instance. If this time limit is exceeded, it is denoted as *t.l.* in the reported results. Detailed results as well as the instances can be found on GitHub[1].

4.2 Computational performance of the natural formulations

In this section, we study and discuss the computational performance of the natural formulations n-ILP$_{\texttt{B}}$ (3.6), n-ILP$_{\texttt{TF}}$ (3.11) and n-ILP$_{\texttt{B+TF}}$ (3.15). The aim of this section is to determine the best Branch-and-Cut algorithm to solve a natural formulation for the MFBP.

4.2.1 Computational performance of the natural formulation n-ILP$_{\texttt{TF}}$

In the case of n-ILP$_{\texttt{TF}}$, the separation problem for target-flow inequalities is $\mathcal{NP}$-hard for fractional solutions (see Proposition 26). Consequently, an efficient separation procedure is crucial. In light of this, we have explored several strategies to separate target-flow inequalities (3.10). The first approach involves separating fractional solutions by solving the exact separation problem (see Model (3.14)). Additionally, alternative methods have been considered, wherein a feasible solution of the separation problem (3.14) is obtained instead of the optimal solution. To this end, we can use an approximation algorithm with a performance guarantee of $\Phi + 1$ designed in Krumke et al. [1999] for the MECFP. The general idea of this approximation algorithm relies on solving a MCCP in the graph. Lastly, we investigate an algorithm to heuristically separate fractional solutions. This algorithm is a heuristic based on successive resolutions of shortest path problems (SPP). The strategies implemented are explained more in detail below.

Integer separation of target-flow inequalities (3.10) One way to implement the Branch-and-Cut algorithm is to separate the target-flow inequalities only for integer LP relaxation points, since modern MIP solvers guarantee that fractional points are cut off by the standard branching procedure strengthen with cutting plane mechanisms. As stated in Proposition 2, this can be done in polynomial time by solving a MFP. However, an efficient separation strategy for fractional points can help improve performance of a Branch-and-Cut based approach.

[1] `https://github.com/ismabentoumi/MaximumFlowBlockerProblem`

Exact separation of target-flow inequalities (3.10) Given a fractional solution $\mathbf{x}^*$ of the RMP, we solve a MECFP where $p_a = x^*_a$ for every arc $a \in A$, as explained in Proposition 26. Let $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ be the optimal solution of this MECFP obtained by solving Model (3.14) using a MIP solver. In case the optimal solution value is strictly smaller than 1, then we have detected a violated target-flow inequality (3.10) and the following cut is added to the RMP:

$$\sum_{a \in A(\hat{\mathbf{y}})} x_a \geq 1.$$

However, as stated in Proposition 26, the MECFP is $\mathcal{NP}$-hard. Accordingly, the exact separation of target-flow inequality may be time consuming.

Approximation separation of target-flow inequalities (3.10) We also explored the possibility of considering a feasible solution of the MECFP instead of the optimal solution. Precisely, instead of optimally solve Model (3.14), we consider an approximation algorithm with performance $R = \Phi + 1$, designed in Krumke et al. [1999]. The general idea of this approximation algorithm consists in solving a MCCP in the graph G where each arc $a \in A$ has a capacity c_a, a lower bound b_a and a flow cost $p_a = \frac{p_a}{c_a}$. For the arc (t, s), values of the capacity and the lower bound are set to $\Phi + 1$. All other arcs $a \in A \setminus \{(t, s)\}$ have the original capacities and a lower bound $b_a = 0$. If the optimal solution value of this MCCP is strictly smaller than 1, then a target-flow inequality is found and it is constructed from the optimal solution $\hat{\mathbf{y}}$ of the MCCP instance. Otherwise, no target-flow inequality has been found. As solving the MCCP is an approximation algorithm for the separation problem of the target-flow inequalities (3.10), it does not provide the optimal solution and accordingly, there might be a target-flow inequality that has not been found. Therefore, the approximation algorithm allows to find a target-flow inequality in a record time. However, when no target-flow inequality is found, it does not ensure that all target-flow inequalities are satisfied by $\mathbf{x}^*$.

Heuristic separation of target-flow inequalities (3.10) **based on shortest paths** Finally, we investigate an algorithm to heuristically generate target-flow inequalities (3.10). Accordingly, we developed an heuristic, based on successive resolutions of shortest path problems, for the MECFP. We recall that a path p is defined as an ordered set of $|p|$ distinct vertices $\{v_1, \cdots, v_{|p|}\}$ such that for all $i \in \{1, \cdots, |p| - 1\}$, (v_i, v_{i+1}) is an arc and we denote by $A(p)$ the set of arcs $\{(v_i, v_{i+1}) \in A, i \in \{1, \cdots, |p| - 1\}\}$. The general idea of the heuristic is to define a flow as a set of $s - t$ paths that will be found iteratively. At initialization, all arcs have a cost x^*_a and a capacity c_a. Note that if $x^*_a = 1$, then $c_a = 0$. At each iteration, for every arc a of the $s - t$ path found, its capacity c_a is reduced by the flow value of the path and its cost is set to 0. The algorithm stops when the total flow value of all paths found exceeds Φ or when no path has been found.

Therefore, we distinguish four different configurations of the Branch-and-Cut algorithm for n-ILP$_{\texttt{TF}}$ based on the separation strategy: *i*) `INT_SEP` *ii*) `EXACT_SEP`, *iii*) `APPROX_SEP`, *iv*) `HEU_SEP`. In the first configuration, we consider only separation of integer infeasible points using the LP interface of `LEMON` to solve the MFP. In the three other

configurations, we consider separation of all LP-Relaxation points (integer and fractional). In EXACT_SEP, fractional LP-Relaxation points are separated using CPLEX MIP Solver applied to the exact separation problem given by model (3.14). In APPROX_SEP, the separation model is replaced by the approximation algorithm presented in Krumke et al. [1999], solved using CPLEX MIP Solver. In HEU_SEP, the heuristic presented previously separates LP relaxations fractional points. This heuristic relies on LEMON LP interface used as a black-box shortest path solver. Note that we consider a time limit of 600 seconds for every resolution of the separation problem and for the total computing time of the Branch-and-Cut algorithm.

In Table 4.1, we compare the four configurations of the Branch-and-Cut algorithm. The set of instances considered is composed by networks of 20 vertices from the SYNTHETIC class. Each row corresponds to 5 instances of this set grouped by the number of arcs $m \in \{40, 54, 86, 116, 146, 208\}$ and value $\lambda \in \{0.2, 0.4, 0.6, 0.8\}$. For each configuration, we report the following values: the number of instances solved to optimality per group (#opt), the average computing time and the maximum computing time in seconds; *t.l.* is reported when the average or the maximum computing time exceeds the time limit set to 600 seconds, the average number of nodes explored in the branching tree (nodes). For INT_SEP, the table reports the average number of constraints generated to separate integer infeasible points (#lazy). For EXACT_SEP, APPROX_SEP and HEU_SEP, the table reports the number of constraints generated to separate fractional infeasible points (#user). It also indicates the average and maximum time to generate one constraint (time user), computed as the total time consumed to solve the separation problem divided by the total number of constraints generated. We recall that separate integer infeasible points is done in polynomial time, as it is equivalent to solve a maximum flow problem.

Table 4.1 shows that INT_SEP solves only 58 instances out of 20 to proven optimality. The algorithm starts facing difficulties for graphs of 20 vertices and 54 arcs. When the size of the graph increases, the number of lazy constraints needed to solve the problem become huge and may be as high as 25498 for m= 86 and $\lambda = 0.2$. Which contributes to increase the number of nodes explored in the branching tree that can reach 18558 for m= 208. This explains the large computing times obtained with INT_SEP. On the other hand, separating fractional unfeasible points is considerably helpful to find the optimal solution of the MFBP. However, the separation strategy has a major impact on efficiency of the Branch-and-Cut algorithm. Indeed, separate LP-Relaxation fractional points through the exact separation problem decreases the computing time and thus increases the number of instances optimally solved (89 instead of 58). However, we can see that this configuration of the algorithm is not as efficient as APPROX_SEP and HEU_SEP. This is mainly due to complexity of the exact separation problem, given by model (3.14). Hence, the average computing time to separate a unique fractional unfeasible point varies between 0.3 and 0.9 seconds for graphs larger than 20 vertices and 116 arcs. As a result, the maximum average number of constraints that can be generated by the exact separation problem before the time limit, is 934 for m= 146 and $\lambda = 0.4$. On the opposite, the average computing time to generate one constraint using

the approximation algorithm or the heuristic algorithm, is nearly zero. This allows to separate a large set of unfeasible fractional points in a short time. In `APPROX_SEP` (resp. `HEU_SEP`), the average number of constraints generated can reach 112286 (resp. 21076) for m= 208 and $\lambda = 0.2$. Finally, we observe that for some instances, `APPROX_SEP` struggles to converge to the optimal solution. For example, for graphs with m= 208 and $\lambda = 0.2$, the average number of unfeasible fractional points separated is 12239, while the number of explored nodes in the branching tree is 837. The computing time is on average 294.4 seconds. In contrast for the same inputs parameters, `HEU_SEP` separates on average 5872 fractional points, while exploring more nodes (1987), with a computing time of 131 seconds. This shows that `HEU_SEP` proposes a better cutting plane strategy in the sense that the Branch-and-Cut algorithm converges to the optimal solution faster. Hence, using the shortest-path based heuristic to solve the separation problem provides the best results for the Branch-and-Cut algorithm. This is probably due to the fact that the heuristic constructs a flow through several paths computed iteratively according to an updated capacity, whereas the approximation algorithm compute a flow at one time.

Therefore, we can conclude that using the shortest-path based heuristic represents the best separation strategy for target-flow inequalities. However, the results provided by this approach remain relatively limited. An alternative approach involves separating only Benders cuts and adding a target-flow inequality each time a Benders cut is found. In what follows, we will see that considering a natural formulation where Benders cuts are separated at every node of the branching tree allows to radically decrease the computing time of the natural formulation.

			INT_SEP					EXACT_SEP							APPROX_SEP							HEU_SEP						
				time		# lazy	nodes		time		# user	time user		nodes		time		# user	time user		nodes		time		# user	time user		nodes
m	λ	#	# opt	avg.	max	avg.	avg.	# opt	avg.	max	avg.	avg.	max	avg.	# opt	avg.	max	avg.	avg.	max	avg.	# opt	avg.	max	avg.	avg.	max	avg.
40	0.2	5	5	0.1	0.3	92.6	41.4	5	0.0	0.1	0.6	0.0	0.0	0	5	0.0	0.0	2.4	0.0	0.0	0	5	0.0	0.0	0.6	0.0	0.0	0
	0.4	5	5	0.0	0.1	61.0	27.4	5	0.0	0.0	0.4	0.0	0.0	0	5	0.0	0.0	1.4	0.0	0.0	0	5	0.0	0.0	0.4	0.0	0.0	0
	0.6	5	5	0.0	0.0	3.2	0	5	0.0	0.0	0.2	0.0	0.0	0	5	0.0	0.0	0.2	0.0	0.0	0	5	0.0	0.0	0.2	0.0	0.0	0
	0.8	5	5	0.0	0.0	2.2	0	5	0.0	0.0	0.0	0.0	0.0	0	5	0.0	0.0	0.0	0.0	0.0	0	5	0.0	0.0	0.0	0.0	0.0	0
54	0.2	5	4	121.1	t.l.	6866.8	3647.6	5	0.1	0.3	3.8	0.0	0.0	0	5	0.0	0.0	11.4	0.0	0.0	3.8	5	0.0	0.0	3.4	0.0	0.0	0
	0.4	5	5	1.0	4.6	654.2	269.4	5	0.1	0.1	2.2	0.0	0.0	0	5	0.0	0.0	6.6	0.0	0.0	4.6	5	0.0	0.0	2.2	0.0	0.0	0
	0.6	5	5	0.0	0.1	42.4	18.8	5	0.0	0.1	1.0	0.0	0.0	0	5	0.0	0.0	2.2	0.0	0.0	1	5	0.0	0.0	1.6	0.0	0.0	0
	0.8	5	5	0.0	0.0	12.2	4.4	5	0.0	0.0	0.2	0.0	0.0	0	5	0.0	0.0	0.2	0.0	0.0	0	5	0.0	0.0	0.2	0.0	0.0	0
86	0.2	5	0	t.l.	t.l.	25498.4	10878	5	7.5	18.6	41.6	0.1	0.2	19.4	5	0.3	0.5	206.6	0.0	0.0	17.6	5	0.1	0.2	42.6	0.0	0.0	27.4
	0.4	5	1	493.4	t.l.	22742.2	13937	5	19.0	56.3	67.8	0.2	0.3	21.2	5	0.3	0.6	181.0	0.0	0.0	29.8	5	0.2	0.4	66.6	0.0	0.0	34.8
	0.6	5	4	136.5	t.l.	8195.4	4735.8	5	10.7	33.1	35.2	0.3	0.3	15.6	5	0.2	0.4	94.4	0.0	0.0	22.6	5	0.1	0.2	36.4	0.0	0.0	21.8
	0.8	5	5	0.1	0.2	59.4	36.6	5	4.0	15.3	12.2	0.2	0.4	4.4	5	0.0	0.1	6.4	0.0	0.0	2	5	0.1	0.1	6.8	0.0	0.0	5
116	0.2	5	0	t.l.	t.l.	21987.2	9331	5	150.3	347.3	330.8	0.5	0.8	38.6	5	5.6	8.3	1987.0	0.0	0.0	212.4	5	1.2	1.7	408.2	0.0	0.0	101.6
	0.4	5	0	t.l.	t.l.	21820.6	12385.8	5	224.9	515.9	567.6	0.3	0.4	38.4	5	9.6	17.2	2769.6	0.0	0.0	220.2	5	1.7	3.1	447.6	0.0	0.0	96.4
	0.6	5	0	t.l.	t.l.	21908.0	15459.2	3	379.5	t.l.	708.2	0.5	0.6	21.6	5	2.9	6.0	1213.2	0.0	0.0	111.6	5	0.9	1.9	178.2	0.0	0.0	61.2
	0.8	5	5	1.2	1.9	803.4	586.4	3	245.0	t.l.	372.8	0.5	0.7	12.8	5	0.1	0.1	20.6	0.0	0.0	4.2	5	0.1	0.3	19.4	0.0	0.0	10
146	0.2	5	0	t.l.	t.l.	20618.4	11699.2	3	324.7	t.l.	629.8	0.5	0.7	37	3	249.4	t.l.	12239.8	0.0	0.0	837.8	5	131.0	554.8	5872.2	0.0	0.0	1987
	0.4	5	0	t.l.	t.l.	20599.2	12793.2	2	483.0	t.l.	934.4	0.5	0.6	46.6	3	271.9	t.l.	13292.0	0.0	0.0	629.2	3	247.8	t.l.	9252.4	0.0	0.0	1473.6
	0.6	5	0	t.l.	t.l.	21705.2	17073.6	3	358.6	t.l.	660.4	0.5	0.6	50.8	4	221.6	t.l.	10081.0	0.0	0.0	517	4	134.5	t.l.	5382.6	0.0	0.0	684
	0.8	5	4	191.7	t.l.	7860.4	6873	4	137.4	t.l.	247.8	0.4	0.6	24.2	5	3.6	15.1	885.2	0.0	0.0	77.4	5	3.7	10.4	347.0	0.0	0.0	75.6
208	0.2	5	0	t.l.	t.l.	16524.2	8168.6	0	t.l.	t.l.	967.0	0.6	0.7	11.8	0	t.l.	t.l.	112286.4	0.0	0.0	444	0	t.l.	t.l.	21076.2	0.0	0.0	1650.4
	0.4	5	0	t.l.	t.l.	16990.2	8728.6	0	t.l.	t.l.	792.6	0.8	0.9	16.2	0	t.l.	t.l.	19170.0	0.0	0.0	413.4	0	t.l.	t.l.	18235.2	0.0	0.0	1014.2
	0.6	5	0	t.l.	t.l.	16324.4	10284	0	t.l.	t.l.	749.4	0.9	1.4	43.4	0	t.l.	t.l.	18483.8	0.0	0.0	466.8	0	t.l.	t.l.	15965.2	0.0	0.0	1045.2
	0.8	5	0	t.l.	t.l.	19019.8	18558.6	1	594.2	t.l.	705.4	0.8	0.9	48	4	221.2	t.l.	9585.2	0.0	0.0	407.6	5	56.8	91.6	2833.8	0.0	0.0	398.6
Total		120	58					89							99							102						

Table 4.1: Performance comparison of the Branch-and-Cut algorithm used to solve the natural formulation n-ILP$_{\texttt{TF}}$ (3.11) on SYNTHETIC instances of 20 vertices grouped by value λ

4.2.2 Computational performance of the natural formulations n-ILP$_{\texttt{B}}$ and n-ILP$_{\texttt{B+TF}}$

In the remainder of this section, we study the computational performance of the natural formulation n-ILP$_{\texttt{B}}$ (3.6) with Benders cuts (3.6b) and the natural formulation enhanced with target-flow inequalities (3.10). This latter formulation is denoted by n-ILP$_{\texttt{B+TF}}$. The aim is to determine the best Branch-and-Cut algorithm to solve the natural formulation for the MFBP.

The natural formulation n-ILP$_{\texttt{B}}$ is solved using a Branch-and-Benders-Cut algorithm where Benders cut are separated at every node of the branching tree and added to the RMP. As explained in Section 3.2.3, this can be done by solving a MCCP, see Model (3.9), using `CPLEX` ILP solver. It is worth noticing that the RMP is initialized with one Benders cut that is associated with the maximum flow in G. This flow is, by definition, a flow of value greater than Φ. We denote by `BENDERS` this Branch-and-Cut algorithm.

In the case of n-ILP$_{\texttt{B+TF}}$, we observed that the performance of the Branch-and-Cut algorithm is negatively impacted by the separation of the target-flow inequalities as the results of n-ILP$_{\texttt{TF}}$ have shown. However, further use of target-flow inequalities can derive from the separation of the Benders cuts. Indeed, it is worth noticing that the separation of target-flow inequalities (3.10) as well as the separation of Benders cuts (3.6b) require finding a flow of value greater than Φ. Consequently, for every Benders cut generated, one may check if the associated target-flow inequality is violated. If so, then this target-flow inequality is added to the RMP. We denote by n-ILP$_{\texttt{B+TF}^*}$ the natural formulation featuring Benders Cuts (3.6b) and target-flow inequalities (3.10), solved by separating Benders Cuts and incorporating target-flow inequalities within the Branch-and-Cut algorithm. This Branch-and-cut algorithm is referred to as `BENDERS_TF`.

Table 4.2 compares performance of the two Branch-and-Cut algorithms presented, i.e., `BENDERS` and `BENDERS_TF`. The instances used for this analysis consist of graphs with $40, 60$ and 100 vertices from the `SYNTHETIC` class with density values of $0.2, 0.4, 0.6$ and 0.8. We consider three values of the target-flow defined by $\lambda = 0.2$, $\lambda = 0.6$ and $\lambda = 0.9$. Each row in the table reports the results of a group of tests conducted on five similar graphs (same vertices and arcs) with distinct blocker costs and capacities on the arcs. The column # reports the total number of instances per row. For each Branch-and-Cut algorithm, we report the following values computed for every group of tests: the number of instances solved to proven optimality (#opt), the average (avg.) and maximum (max) computing time (time) in seconds and the average number of Benders constraints generated to separate integer and fractional infeasible points (#benders). For `BENDERS_TF`, we also report the average number of target-flow inequalities generated (#TF). Table 4.2 illustrates that `BENDERS_TF` achieves better computational performance than `BENDERS`. Indeed, `BENDERS_TF` manages to solve 159 instances out of 180 against 156 for `BENDERS`. This result is also supported by the consistently lower computing time of `BENDERS_TF` compared to `BENDERS`. Furthermore, we observe that incorporating target-flow inequalities into the Branch-and-Cut algorithm reduces the number of Benders cuts generated and allows us to reach the optimal solution faster. As an illustrative example, for graphs of 40 vertices with a density of 0.6, when using the algorithm `BENDERS`, the average number of Benders cut generated is approximately

232. By adding 20 target-flow inequalities, this number reduces to 165 (slightly more than half of the original amount). For graphs of 60 vertices with a density of 0.6, using the algorithm BENDERS_TF leads to a reduction of 423 Benders cuts while adding 95 target-flow inequalities. In addition, for graphs of 40 and 60 vertices, the average time spent on the separation of Benders cuts is approximately 0.01 seconds. For graphs of 100 vertices, this value averages 0.05 seconds, which is still relatively short. However, it is important to consider that for some instances, a significant number of Benders cuts need to be generated to reach the optimal solution. This contributes to an overall increase in the total computing time. It is worth noticing that all Benders cuts are not generated in the root node of the branching tree. Moreover, the separation of Benders cuts is always performed with a violation tolerance of 10^{-6}. To conclude, Table 4.2 shows that the most efficient algorithm to solve the natural formulation for the MFBP is BENDERS_TF, which corresponds to a Branch-and-Cut algorithm that separates Benders cuts (3.6b) at every node of the branching tree and incorporates target-flow inequalities when violated. In the remainder of this paper, we consider this algorithm to solve a natural formulation for the MFBP and it is denoted by n-ILP.

			BENDERS				BENDERS_TF				
				time		#benders		time		#benders	#TF
n	$d(G)$	#	#opt	avg.	max	avg.	#opt	avg.	max	avg.	avg.
40	0.2	15	15	0	0.1	19	15	0	0.10	18	2
	0.4	15	15	0.12	0.53	54	15	0.21	0.51	45	8
	0.6	15	15	1.19	7.9	232	15	0.86	4.21	165	20
	0.8	15	15	3.13	12.77	349	15	3	10.91	341	43
60	0.2	15	15	0.49	3.97	91	15	0.42	4.12	79	18
	0.4	15	15	4.64	35.72	329	15	5.37	40.59	314	84
	0.6	15	14	91	t.l.	1441	15	40.56	169.91	1018	95
	0.8	15	14	147.65	t.l.	1399	14	124.55	t.l.	1258	91
100	0.2	15	13	108.42	t.l.	1395	14	77.88	t.l.	1168	282
	0.4	15	11	181.84	t.l.	1142	11	184.25	t.l.	1033	93
	0.6	15	8	317.21	t.l.	1787	8	321.15	t.l.	1526	105
	0.8	15	6	363.14	t.l.	1229	7	351.33	t.l.	1207	142
Total		180	156				159				

Table 4.2: Performance comparison of the Branch-and-Cut algorithms for n-ILP$_{\texttt{B}}$ and n-ILP$_{\texttt{B+TF}^*}$, i.e. BENDERS and BENDERS_TF, on SYNTHETIC instances

4.2.3 Comparison between the natural formulation and the state-of-the-art technique

We now compare the performance of the natural formulations against the direct use of a state-of-the-art exact solver for mixed-integer bilevel linear programs (MIBLPs),

available online[2]. This solver is associated with the work of Fischetti and Ljubic [2017], where the authors propose a proof-of-concept exact MIBLP solver based on a Branch-and-Cut MILP approach, strengthened with new families of cuts, namely intersection cuts (see Fischetti et al. [2018]) and hypercube intersection cuts. Additionally, the solver incorporates a preprocessing procedure and many other features including heuristics, propagations, multi-threading support, and LP parametrization. For our experiments, we use the setting MIX++, which represents the robust default setting for the Branch-and-Cut algorithm employed by the solver. MIX++ separates intersection cuts at each separation call using the separation procedure described in Fischetti and Ljubic [2017] and the extended bilevel-free set of Xu [2012]. It also applies preprocessing to the follower problem. By the tests made with the natural formulations, the solver is used in a single-thread mode. This resolution method is denoted by `BISOLVER`.

Table 4.3 compares performance of the bilevel solver, i.e, `BISOLVER`, and the two Branch-and-Cut algorithms for solving n-ILP$_{\texttt{B}}$ and n-ILP$_{\texttt{B+TF}^*}$, i.e., `BENDERS` and `BENDERS_TF`. The instances used for this analysis are the same as the one used in Table 4.2.

Table 4.3 illustrates that the two Branch-and-Cut algorithms proposed for n-ILP$_{\texttt{B}}$ and n-ILP$_{\texttt{B+TF}^*}$ outperform the direct use of the bilevel solver. Indeed, with `BISOLVER`, only 28 instances out to 180 have been solved to proven optimality, which constitutes less than 16% of the instances. The time limit is reached even for instances with 40 vertices and a density of 0.4. In the meanwhile, the two natural formulations begin to encounter challenges for graphs with 60 vertices and a density of 0.6. Remark that, at this stage, the bilevel solver fails to solve any instance.

			BENDERS				BENDERS_TF					BISOLVER		
				time		#B		time		#B	#TF		time	
n	$d(G)$	#	#opt	avg.	max	avg.	#opt	avg.	max	avg.	avg.	#opt	avg.	max
40	0.2	15	15	0	0.1	19	15	0	0.10	18	2	15	4.53	19.31
	0.4	15	15	0.12	0.53	54	15	0.21	0.51	45	8	4	495.17	t.l.
	0.6	15	15	1.19	7.9	232	15	0.86	4.21	165	20	1	592.99	t.l.
	0.8	15	15	3.13	12.77	349	15	3	10.91	341	43	0	t.l.	t.l.
60	0.2	15	15	0.49	3.97	91	15	0.42	4.12	79	18	8	t.l.	t.l.
	0.4	15	15	4.64	35.72	329	15	5.37	40.59	314	84	0	t.l.	t.l.
	0.6	15	14	91	t.l.	1441	15	40.56	169.91	1018	95	0	t.l.	t.l.
	0.8	15	14	147.65	t.l.	1399	14	124.55	t.l.	1258	91	0	t.l.	t.l.
100	0.2	15	13	108.42	t.l.	1395	14	77.88	t.l.	1168	282	0	t.l.	t.l.
	0.4	15	11	181.84	t.l.	1142	11	184.25	t.l.	1033	93	0	t.l.	t.l.
	0.6	15	8	317.21	t.l.	1787	8	321.15	t.l.	1526	105	0	t.l.	t.l.
	0.8	15	6	363.14	t.l.	1229	7	351.33	t.l.	1207	142	0	t.l.	t.l.
Total		180	156				159					28		

Table 4.3: Performance comparison of the Branch-and-Cut algorithms `BENDERS` and `BENDERS_TF` and the bilevel solver (`BISOLVER`) on `SYNTHETIC` instances.

[2] `https://msinnl.github.io/pages/bilevel.html`

4.3 Comparison of the effectiveness of the natural and the compact formulation

In this section, we compare the computational performance of our best Branch-and-Cut algorithm for the natural formulation n-ILP, i.e., `BENDERS_TF` against c-ILP solved by `CPLEX` MIP Solver.
Due to the combinatorial nature of integer programs, the latest versions of `CPLEX` perform some operations to improve the branching algorithm. This is done automatically in the default setting. However, `CPLEX` has many parameters that allow the user to customize resolution of the problem. We will study the effectiveness of these parameters in the next section.

4.3.1 Resolution of the compact formulation

In the following, we identify two settings that may affect performance of c-ILP. The first one decides whether CPLEX applies presolve during preprocessing. The presolve process performs several reductions to eliminate some variables and thus reduce the size of the problem. The second one sets the upper limit on the number of cutting plane passes performed by CPLEX in the root node. Therefore, we consider four variants for c-ILP, that we detail below.

1. c-ILP `default`: In this variant, all `CPLEX` parameters are set to default values
2. c-ILP `presolve`: In this variant, `CPLEX` does not use presolve and all other parameters are set to default values
3. c-ILP `cuts`: In this variant, `CPLEX` does not perform any cutting plane passes in the root node and all other parameters are set to default values
4. c-ILP `presolve cuts`: In this variant, `CPLEX` uses neither presolve nor cutting plane strategy and all other parameters are set to default values.

For this comparison, we consider `SYNTHETIC` instances. As in the previous experiments, a time limit of 600 seconds is set for each run. The results of these tests are reported in Table 4.4. In this table, we compare results of n-ILP and all configurations of c-ILP on instances of the `SYNTHETIC` class with $n \in \{50, 100, 300, 500\}$. For a given number of vertices, each row corresponds to 55 instances grouped by the density $d(G) \in \{0.2, 0.4, 0.6, 0.8\}$. We consider different target-flow values defined by $\lambda \in \{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.999\}$. For each formulation, we report the following values: the number of instances optimally solved on each group ("#opt"), the average computing time, the maximum computing time and the number of nodes in the branching tree. As previously, *t.l.* is reported if the computing time exceeds the time limit. Table 4.4 directly shows that c-ILP outperforms n-ILP independently of the input graph or `CPLEX` features. Indeed, we first focus on instances of 50 and 100 vertices. All configurations of c-ILP finds the optimal solution within a very short time (from 0.0 to 0.4 seconds). In reverse, even for this category of instances, n-ILP struggles to solve

the MFBP. For example, for graphs of 50 vertices with a density 0.2, n-ILP finds the optimal solution of 15 out of 55 instances only. Note that the number of nodes in the branching tree increases with the size of the graph and can reach 1086.6, while c-ILP always finds the optimal solution in the root node. Increase of the number of vertices (e.g., $n = 100$) leads to a fall in efficiency of the formulation; only 5 instances solved for each density value. We now investigate performance of the two formulations for graphs of 300 and 500 vertices. As expected, n-ILP is not able to solve at optimality any instance. In addition, for instances with a size greater than 300 vertices with a density 0.2, the number of nodes in the branching tree is equal to 0. This shows that even finding a feasible solution in the root node is time consuming. However, it is interesting to see that for this category of instances, performance of c-ILP varies according to `CPLEX` features. First, using default `CPLEX` parameters provides the best results for c-ILP. In fact, c-ILP `default` solves to proven optimality all the 880 instances in a reasonable time; the maximum computing time for an instance of 500 vertices with a density 0.8 is 237.6 seconds. Second, the presolve parameter impact the most performance of c-ILP. By performing any preprocessing, efficiency of c-ILP significantly deteriorates. This can be seen through the average computing time that have increased; from 1 second to 25 seconds for instances of 300 vertices with a density 0.2. This can also be seen though the number of instances solved to proven optimality, that have decreased; from 880 to 858. The number of cutting plane passes also slightly affects performance of c-ILP; 3 instances from the 880 instances have not been solved to proven optimality with c-ILP `cuts`. Finally, using neither presolve nor cutting plane passes is the worst configuration of c-ILP with 857 instances solved to proven optimality out of 880. Remark that for c-ILP `presolve`, c-ILP `cuts` and c-ILP `presolve cuts`, the number of nodes in the branching tree radically increases; from 0 to respectively 3249, 3634 and 4173 for instances of 300 vertices with a density 0.2. This shows the effect of `CPLEX` improvements for c-ILP, especially in the root node.

In conclusion, c-ILP `default` holds the best configuration to solve the MFBP through the compact formulation (3.18). In the rest of this computational study, we consider c-ILP `default` configuration for c-ILP.

			c-ILP `default`				c-ILP `presolve`				c-ILP `cuts`				c-ILP `presolve cuts`				n-ILP			
				time		nodes		time		nodes		time		nodes		time		nodes		time		nodes
n	$d(G)$	#	# opt	avg.	max	avg.	# opt	avg.	max	avg.	# opt	avg.	max	avg.	# opt	avg.	max	avg.	# opt	avg.	max	avg.
50	0.2	55	55	0.0	0.0	0	55	0.0	0.0	0.0	55	0.0	0.0	0.0	55	0.0	0.0	0.0	15	450.1	t.l.	494.9
	0.4	55	55	0.0	0.1	0	55	0.0	0.1	0.0	55	0.0	0.0	0.0	55	0.0	0.1	0.0	5	545.5	t.l.	243.2
	0.6	55	55	0.0	0.2	0	55	0.0	0.2	0.0	55	0.0	0.1	0.0	55	0.0	0.2	0.0	5	545.6	t.l.	592.4
	0.8	55	55	0.0	0.1	0	55	0.0	0.1	0.0	55	0.0	0.1	0.0	55	0.0	0.1	0.0	5	545.6	t.l.	1086.6
100	0.2	55	55	0.1	0.1	0	55	0.1	0.1	0.0	55	0.1	0.1	0.0	55	0.1	0.1	0.0	5	545.6	t.l.	1117.0
	0.4	55	55	0.1	0.2	0	55	0.1	0.2	0.0	55	0.1	0.2	0.0	55	0.1	0.2	0.0	5	546.5	t.l.	1583.2
	0.6	55	55	0.2	0.3	0	55	0.2	0.2	0.0	55	0.2	0.2	0.0	55	0.2	0.3	0.0	5	547.0	t.l.	1417.0
	0.8	55	55	0.3	0.4	0	55	0.3	0.4	0.0	55	0.3	0.4	0.0	55	0.3	0.4	0.0	5	t.l.	t.l.	1036.4
300	0.2	55	55	1.1	1.8	0	53	25.0	t.l.	3249.8	53	24.7	t.l.	3634.2	53	24.8	t.l.	4173.6	0	t.l.	t.l.	179.7
	0.4	55	55	3.3	5.0	0	55	13.8	28.9	0.0	55	3.4	5.5	0.0	55	11.4	23.5	0.0	0	t.l.	t.l.	0.0
	0.6	55	55	6.5	10.9	0	55	31.1	70.7	0.0	55	6.6	11.2	0.0	54	34.1	t.l.	607.2	0	t.l.	t.l.	0.0
	0.8	55	55	9.2	16.1	0	55	63.5	146.2	0.0	55	9.4	17.6	0.0	54	62.4	t.l.	353.4	0	t.l.	t.l.	0.0
500	0.2	55	55	6.1	11.1	0	54	39.1	t.l.	583.9	54	21.9	t.l.	645.0	54	39.2	t.l.	590.7	0	t.l.	t.l.	0.0
	0.4	55	55	17.2	65.4	0	55	101.6	280.8	0.0	55	14.5	29.8	0.0	55	100.6	280.5	0.0	0	t.l.	t.l.	0.0
	0.6	55	55	33.2	74.0	0	52	290.7	t.l.	0.0	55	23.3	48.8	0.0	52	294.1	t.l.	0.0	0	t.l.	t.l.	0.0
	0.8	55	55	51.0	237.6	0	39	457.3	t.l.	0.0	55	35.3	91.1	0.0	40	462.2	t.l.	0.0	0	t.l.	t.l.	0.0
Total		880	880				858				877				857				50			

Table 4.4: Performance comparison between all configurations of c-ILP and n-ILP on `SYNTHETIC` instances

4.3.2 Comparison between the natural and the compact formulation

We now consider the same results presented in Table 4.4, focusing only on c-ILP and n-ILP and considering three values of the target-flow defined by $\lambda \in \{0.2, 0.6, 0.9\}$, in order to highlight the difference in performance of the two models. Table 4.5 shows, as previously, that c-ILP outperforms n-ILP. For the smallest graphs ($n = 50$), c-ILP solves all instances in record time; less than 0.1 seconds for all densities. For the same group of instances, n-ILP manages to solve all instances in a reasonable time, i.e., at most 27.88 seconds for a density of 0.8. However, for larger graphs, the performance spread between the two formulations is exacerbated. More precisely, we observe that for graphs of 100 vertices with a density of 0.4, n-ILP fails to solve three instances to proven optimality. This number increases to seven instances for a density $d(G)$ equal to 0.8. Subsequently, for graphs of 300 vertices and a density larger than 0.2, none of the instances were solved to proven optimality within the time limit. In addition, for the natural formulation n-ILP, the number of nodes explored in the branching tree tends to increase as the size of the graph grows. In particular, for graphs with 50 vertices and a density of 0.8, the maximum number of explored nodes reaches an approximate value of 190 while for graphs with 100 vertices, this number averages around 530. For larger instances, such as graphs of 300 vertices with a density greater than or equal to 0.6, no nodes are explored within the time limit. This observation highlights that the linear relaxation of n-ILP requires a substantial amount of time. Furthermore, based on the results from previous instances, it can be inferred that the time invested in the linear relaxation does not contribute to an overall reduction in the computing time of the Branch-and-Cut algorithm for n-ILP. In contrast to n-ILP, c-ILP successfully solves all instances with 100 and 300 vertices at the root node, achieving a maximum computing time of 2.13 seconds. As explained previously, it is worth noticing that the high performance of c-ILP comes essentially from enhancements made by the solver, especially when performing presolve and adding cuts to the model. This can be explained by the structure of the compact formulation c-ILP. More precisely, in Section 3.3, it has been shown that the solution of the MFBP is contained in a cut of the graph and constraints (3.18c) and(3.18d) of c-ILP are constraints defining a cut. Furthermore, constraint (3.18b) belongs to a well-known optimization problem, namely the knapsack problem (see Garey and Johnson [1979] for further details). This shows that solution of the MFBP can be found by solving a knapsack problem in a specific cut of the graph. Hence, constraints of c-ILP are familiar to `CPLEX`, that will be able to perform good preprocessing and general improvements.

			n-ILP				c-ILP			
				time		nodes		time		nodes
n	$d(G)$	#	# opt	avg.	max	avg.	# opt	avg.	max	avg.
50	0.2	15	15	0.02	0.07	7	15	0.01	0.01	0
	0.4	15	15	0.18	0.61	37.53	15	0.03	0.06	0
	0.6	15	15	0.7	2.65	71.33	15	0.04	0.05	0
	0.8	15	15	3.41	27.88	189.67	15	0.05	0.08	0
100	0.2	15	14	77.8	t.l.	214	15	0.05	0.08	0
	0.4	15	11	184.2	t.l.	462.87	15	0.06	0.08	0
	0.6	15	8	321.1	t.l.	263.73	15	0.08	0.09	0
	0.8	15	7	351.3	t.l.	530.87	15	0.11	0.21	0
300	0.2	15	5	469.73	t.l.	251.6	15	0.26	0.41	0
	0.4	15	0	t.l.	t.l.	2.27	15	0.53	0.92	0
	0.6	15	0	t.l.	t.l.	0	15	0.94	1.51	0
	0.8	15	0	t.l.	t.l.	0	15	1.48	2.13	0
Total		180	105				180			

Table 4.5: Performance comparison between c-ILP and n-ILP on `SYNTHETIC` instances

Table 4.6 presents a comparison of c-ILP and n-ILP applied to instances from the `GRID` class and the `REAL-NETWORKS` class. The first six rows correspond to instances of the `GRID` class grouped by the size, written as $n_1 \times n_2$. We count 17 grid graphs with distinct blocker costs and capacities including two grid graphs of size 10×20 and 30×60, one grid graph of size 20×160 and 80×40, five grid graphs of size 20×40 and six grid graphs of size 40×80. The last two rows correspond to instances of the `REAL-NETWORKS` class grouped by the type, i.e., `Type A` or `Type B`. As for the `SYNTHETIC` instances, Table 4.6 demonstrates that c-ILP outperforms n-ILP by several orders of magnitude. For small `GRID` instances (grids of size 10×20, 20×40 and 30×60), c-ILP reaches an optimal solution in less than 2 seconds, while n-ILP takes an average of approximately 334 seconds for grids of size 20×40 and around 529 seconds for grids of size 30×60. The same performance disparities can be observed for the remaining instances in the `GRID` class. For instances of `Type A` and `Type B` from the `REAL-NETWORKS` class, n-ILP requires an average computing time of 259 seconds and 318 seconds, respectively, while c-ILP solves all `REAL-NETWORKS` instances in less than 10 seconds ($\approx$ 9.62 seconds). Moreover, it is worth noticing that c-ILP manages to solve all the 93 instances considered for this study, while n-ILP is only able to solve 41 instances to proven optimality (slightly more than 44%). Finally, we observe that on average, for `GRID` instances and `REAL-NETWORKS` instances, c-ILP finds an optimal solution at the root node, while n-ILP tends to explore a much larger number of nodes (up to 4630.80). Therefore, as for the `SYNTHETIC` instances (see Table 4.5), c-ILP finds an optimal solution of the MFBP very quickly compare to n-ILP and in most cases at the root node, resulting in a better global computational performance. Table 4.6

reveals that our best Branch-and-Cut algorithm for the natural formulation n-ILP does not solve all instances from Royset and Wood [2007].

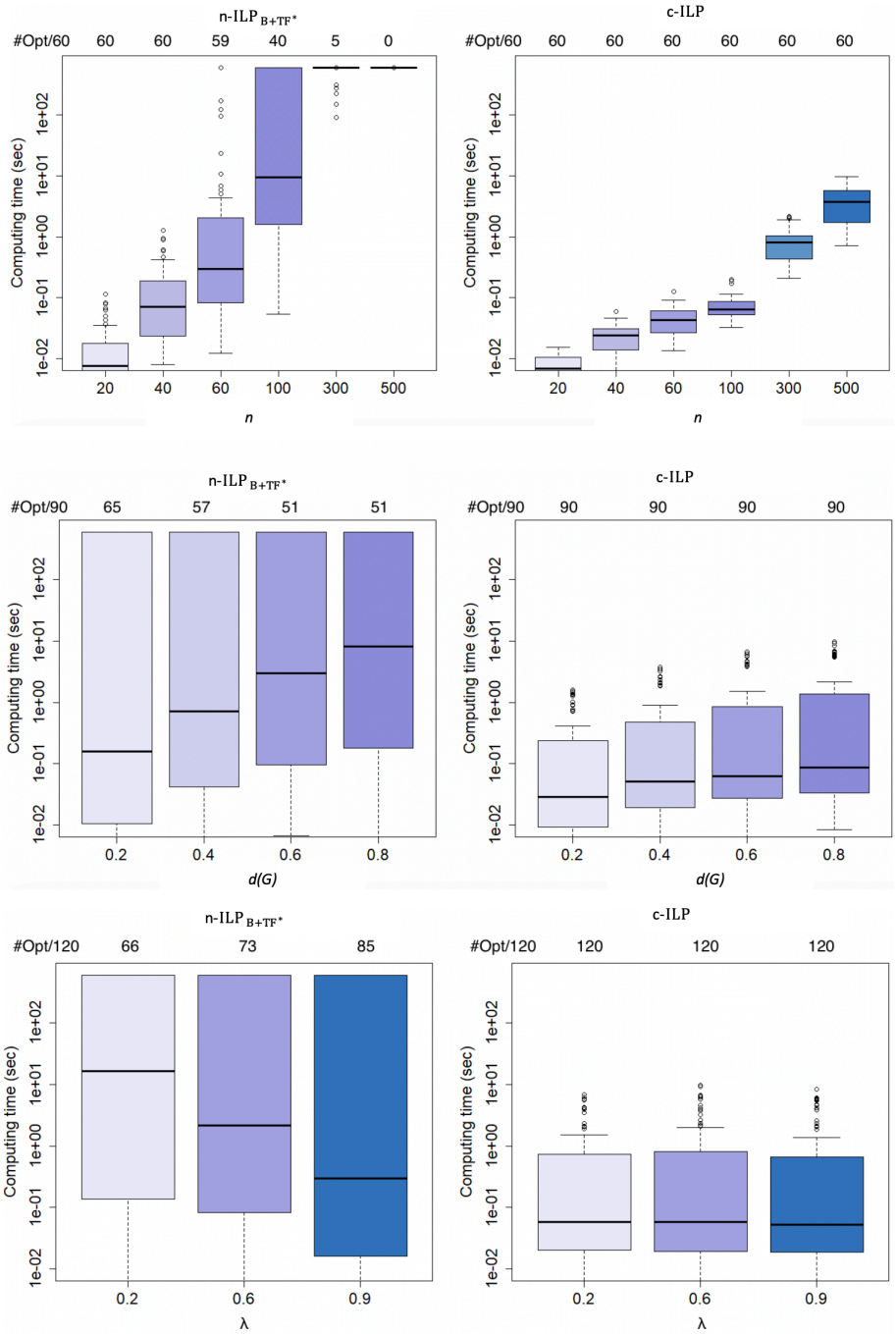
			n-ILP				c-ILP			
				time		nodes		time		nodes
Class	Size\Type	#	# opt	avg.	max	avg.	# opt.	avg.	max	avg.
GRID	10 × 20	6	6	8.55	25.86	1643.17	6	0.07	0.09	0.00
	20 × 160	3	1	542.46	t.l.	1621.00	3	3.99	5.89	0.00
	20 × 40	15	8	334.17	t.l.	4630.80	15	0.37	0.67	0.00
	30 × 60	6	1	529.97	t.l.	3247.00	6	1.37	2.81	0.00
	40 × 80	18	0	t.l.	t.l.	1466.33	18	4.12	8.11	0.00
	80 × 40	3	1	462.97	t.l.	543.33	3	3.24	3.28	0.00
REAL-NETWORKS	Type A	21	13	259.97	t.l.	2355.76	21	1.65	9.62	0.00
	Type B	21	11	318.57	t.l.	2759.90	21	2.06	8.14	0.48
Total		93	41				93			

Table 4.6: Performance comparison between c-ILP and n-ILP on GRID and REAL-NETWORKS instances

In Figure 4.2, we discuss more precisely the impact of graph-structural features and flow-blocker features on the performance of c-ILP and n-ILP (corresponding to Formulation n-ILP$_{\text{B+TF}^*}$). Figure 4.2 gives the computing time boxplots of the two formulations grouping the instances from the SYNTHETIC class by the number of vertices in the graph n, the density $d(G)$ and value λ for the target-flow. We graphically show the time spent by each formulation through their quartiles. The lines extending vertically from the boxes indicate the variability outside the upper and lower quartiles. Above the upper quartile, the outliers are plotted as individual points. The y-axis is the computing time (in logarithmic scale) and the x-axis represents the group of instances. On the top part of the figure, we report, for each group, the total number of instances solved to proven optimality (#Opt). In the top-left corner, we report #Opt/k, where k is replaced by the total number of instances in each group of tests for a given parameter (n, $d(G)$, and λ). These boxplots provide evidence that c-ILP is the best-performing model to solve the MFBP, independently of the size of the graph and the target-flow. However, for both formulations, we observe that the computing time increases as the size of the network grows. Indeed, the number of vertices and the density of the graph are directly correlated to the number of variables and constraints of the two models (see (3.18) for the compact formulation c-ILP and (3.15) for the natural formulation n-ILP). As an illustrative example, for graphs of 500 vertices, the natural formulation does not reach an optimal solution within the time limit. Regarding graph densities, we observe that n-ILP successfully solves 65 instances with a density of 0.2, which decreases to 57 and 51 for greater densities. On the other hand, for graphs of up to 500 vertices, c-ILP

reaches an optimal solution with a very reasonable computing time, i.e., almost half of the time limit. Finally, Figure 4.2 illustrates that n-ILP is more efficient for large values of the target-flow. For example for $\lambda = 0.9$, 85 instances out of 120 have been solved to proven optimality within the time limit, while for $\lambda = 0.2$, n-ILP manages to solve only 66 instances. This is consistent with the constraints of the natural formulation n-ILP$_{\texttt{B+TF*}}$. Indeed, due to the structure of Benders cuts (3.6b), when increasing the value of the target-flow, the left member decreases, which makes the problem easier to solve. In conclusion, Table 4.5, Table 4.6 and Figure 4.2 demonstrate that c-ILP outperforms n-ILP, regardless of the graph-structural features or the flow-blocker features.

Figure 4.2: Computing time boxplots of n-ILP (n-ILP$_{\texttt{B+TF*}}$) and c-ILP on `SYNTHETIC` instances



Since n-ILP$_{\texttt{B+TF}^*}$ (3.15) and c-ILP (3.18) share a family of variables, i.e., $\boldsymbol{x}$ variables which correspond to the blocker decision variables, the target-flow inequalities (3.10) and the Benders cuts (3.6) are valid inequalities for the compact formulation. These inequalities can be employed to further enhance the performance of c-ILP. Hence, we performed some experiments on instances from the SYNTHETIC class to evaluate the impact of these constraints in the compact formulation. However, the addition of these cuts does not yield any noticeable computational enhancements, even in larger instances. This can be explained by the optimizations performed by CPLEX ILP solver that offers high computing performance.

4.4 Gaps

In this section, we are interested in the quality of the LP relaxation of n-ILP$_{\texttt{B}}$, n-ILP$_{\texttt{TF}}$, n-ILP$_{\texttt{B+TF}}$ and c-ILP. The LP relaxation of c-ILP is solved using the LP solver of CPLEX. Solution of the LP relaxation of n-ILP$_{\texttt{B}}$ and n-ILP$_{\texttt{B+TF}}$ involves separating the Benders cuts by solving the separation problem, i.e., Model (3.9) (see Section 3.2.3). In the LP relaxation of n-ILP$_{\texttt{TF}}$ and n-ILP$_{\texttt{B+TF}}$, the target-flow inequalities are separated by solving Model (3.14). In the case of n-ILP$_{\texttt{B+TF}}$, a solution of the LP relaxation is found when all Benders cuts and all target-flow inequalities are separated.

This study focuses on instances derived from the SYNTHETIC class, including graphs of 20, 40 and 50 vertices with a density of 0.2, 0.4, 0.6 and 0.8. To broaden our analysis, we extend the range of λ values. Specifically, we consider 9 values of λ ranging from 0.1 to 0.9 with a step size of 0.1. The results are reported in Table 4.7, where each row presents tests conducted on 45 graphs sharing the same number of vertices and density. For each formulation, we report the average and maximum value of *lp gap*, the optimality gap of the LP relaxation. For each instance, lp gap is computed according to the optimal solution value *opt* as $100 \times \frac{opt - lp_{val}}{opt}$, where lp_{val} is the value of the linear programming relaxation of the corresponding formulation. For n-ILP$_{\texttt{B}}$, n-ILP$_{\texttt{TF}}$ and n-ILP$_{\texttt{B+TF}}$, we also report the average and maximum number of cuts generated to solve the LP relaxation (#B for Benders cuts and #TF for target-flow inequalities). Finally, for n-ILP$_{\texttt{TF}}$ and n-ILP$_{\texttt{B+TF}}$, we report the number of instances solved for which the lp relaxation solution has been found within the time limit set to 1800 seconds (#optLP). It is worth noticing that instances for which the LP relaxation has not been found are not considered in the computation of lp gap, #B and #TF.

For n-ILP$_{\texttt{B}}$ and c-ILP, all LP relaxation values are computed within the time limit. Specifically, computing the LP relaxation of c-ILP takes only seconds. In the case of n-ILP$_{\texttt{B}}$, the average computing time does not exceed 600 seconds, although it may be greater than that of c-ILP. On the other hand, the computation of the LP relaxation of n-ILP$_{\texttt{TF}}$ is time-consuming. Indeed, only 278 instances out of 540 (less than half of the instances) were successfully solved. The number of instances solved decreases with the size of the graph, as evidenced in the table, where only 2 instances were solved for graphs featuring 50 vertices and a density of 0.8. This is due to the $\mathcal{NP}$-hardness of the separation problem (3.14) for the target-flow inequalities. In the same line, for n-ILP$_{\texttt{B+TF}}$, the separation of target-flow inequalities results in an increase in computing

time compared to n-ILP$_{\texttt{B}}$, and some instances have not been solved within the time limit (31 instances).

Section 3.2.6, shows that n-ILP$_{\texttt{B}}$ and n-ILP$_{\texttt{TF}}$ do not dominate each other. In other words, there are instances for which the LP relaxation of n-ILP$_{\texttt{B}}$ is strictly stronger than the one of n-ILP$_{\texttt{TF}}$, and vice versa. This is highlighted by Table 4.7, where the average and maximum lp gap values computed with n-ILP$_{\texttt{B}}$ can either be smaller or greater than those computed with n-ILP$_{\texttt{TF}}$ for a specific group of instances. As an illustrative example, for the smallest instances of 20 vertices with a density 0.2, the maximum lp gap computed with n-ILP$_{\texttt{B}}$ reaches 90, while for the same group of instances, the maximum lp gap of n-ILP$_{\texttt{TF}}$ is equal to 84.44. On the contrary, with an equal number of vertices and a higher density (0.6), n-ILP$_{\texttt{B}}$ exhibits a smaller maximum lp gap (84.67) compared to that obtained with n-ILP$_{\texttt{TF}}$ (90.00). It is worth noticing that the small values of lp gap computed by n-ILP$_{\texttt{TF}}$ for large instances $(2.78, 2.76, 0.00)$ are due to the exclusion of unsolved instances from the computations.

In Section 3.3.3, we demonstrate that c-ILP dominates n-ILP$_{\texttt{B}}$ in terms of LP relaxation (see Proposition 7). Table 4.7 also supports this result. Specifically, the average value and the maximum value of lp gap for c-ILP are always smaller than the one obtained with n-ILP$_{\texttt{B}}$. We are now interested in measuring the extent to which c-ILP outperforms n-ILP$_{\texttt{B}}$. For the instances considered, the difference between the average values of lp gap is relatively close. As an illustrative example, when considering graphs with 40 vertices and a density of 0.2, the LP relaxation quality provided by c-ILP surpasses that of n-ILP$_{\texttt{B}}$ by approximately 11%. Regarding the maximum computed values of lp gap, this difference becomes more noticeable. Specifically, for graphs with 20 vertices and a density of 0.6, the maximum LP gap value peaks at 70.14 for n-ILP$_{\texttt{B}}$, whereas this value stands at 58.57 for c-ILP ($\approx 17\%$ better).

Finally, this table illustrates that c-ILP and n-ILP$_{\texttt{B+TF}}$ are not directly comparable in terms of LP relaxation, as shown in Proposition 7. Indeed, on average, the lp gap of n-ILP$_{\texttt{B+TF}}$ consistently outperforms the average lp gap value computed by c-ILP. However, it is worth noticing that in specific instances, the maximum lp gap is smaller for c-ILP. For example, when considering graphs with 20 vertices and a density of 0.6, the maximum LP gap value for n-ILP$_{\texttt{B+TF}}$ is 70.14, while the corresponding value for c-ILP is 58.57.

4.5 Testing the limits of the compact formulation

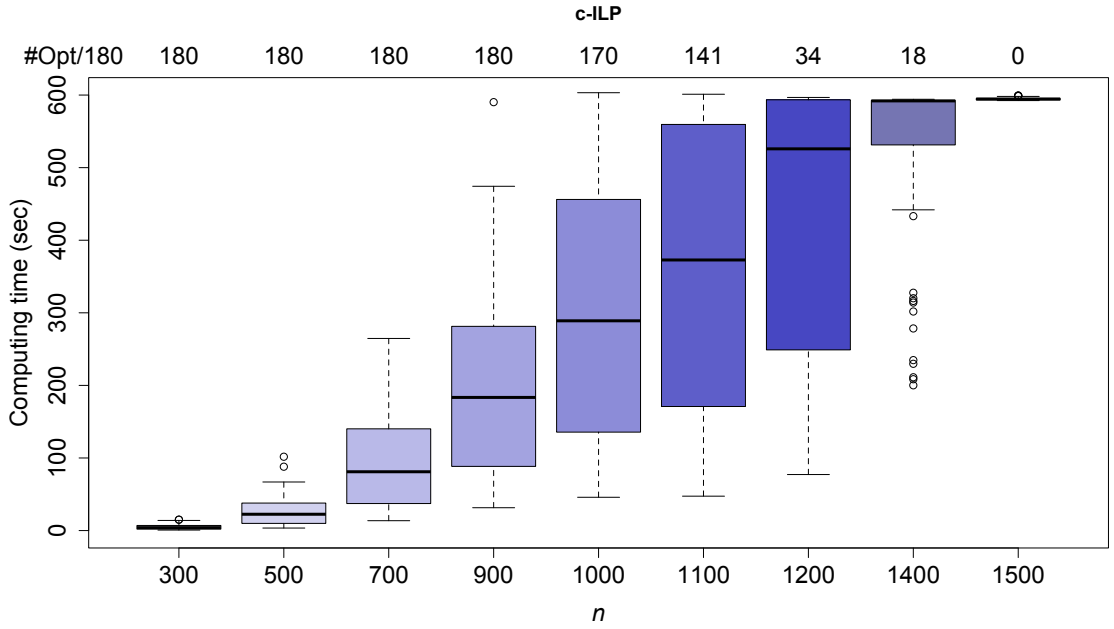
In this section, we test the efficiency of c-ILP in solving large-scale graphs. More precisely, we are seeking to achieve the limits of the compact formulation (3.18). By doing so, we can determine the maximum size of instances that can be solved to proven optimality within a fixed amount of CPU time limit of 600 seconds, similar to the previous experiments. To this end, we include a more extensive set of instances derived from the `SYNTHETIC` class, including graphs with a wide range of values for the number of vertices n. As in our previous experiments, we focus on four specific arc densities: 0.2, 0.4, 0.6,

			n-ILP$_{\texttt{B}}$			n-ILP$_{\texttt{TF}}$				n-ILP$_{\texttt{B+TF}}$					c-ILP	
			#B	lp gap			#TF	lp gap			#B	#TF	lp gap		lp gap	
n	$d(G)$	#	avg.	avg.	max.	#optLP	avg.	avg.	max.	#optLP	avg.	avg.	avg.	max.	avg.	max.
20	0.2	45	1.0	50.0	90.0	45	21.6	76.2	84.4	45	1.0	51.3	13.3	90.0	41.9	88.9
	0.4	45	7.0	29.1	84.7	45	542.0	53.1	90.0	45	53.2	77.5	4.7	71.6	25.4	80.0
	0.6	45	9.8	20.0	70.1	42	1530.5	63.8	87.2	45	116.4	127.0	6.3	70.1	18.3	58.6
	0.8	45	26.6	22.8	63.9	35	2641.1	59.4	80.4	45	166.9	178.6	6.0	46.3	19.8	61.5
40	0.2	45	22.3	26.7	80.5	41	2072.3	56.8	89.3	45	226.3	249.8	8.8	80.5	23.8	76.2
	0.4	45	75.8	20.6	65.7	13	997.2	25.6	80.7	44	610.4	621.1	4.1	48.7	18.0	64.3
	0.6	45	72.7	10.1	30.0	12	818.2	14.0	90.0	44	355.0	305.5	3.5	27.3	9.1	26.5
	0.8	45	177.5	11.0	42.7	6	684.8	2.8	16.7	38	357.5	353.9	4.8	37.6	9.9	40.0
50	0.2	45	50.3	26.2	80.8	23	1301.5	36.2	84.9	45	520.3	529.0	4.8	80.8	24.4	76.4
	0.4	45	153.5	22.1	70.5	10	1462.6	2.8	14.3	36	593.5	545.1	8.7	70.5	19.6	66.7
	0.6	45	127.6	10.3	57.5	4	608.5	0.0	0.0	37	390.5	392.4	5.9	57.5	9.2	55.0
	0.8	45	355.1	9.2	37.4	2	793.5	0.0	0.0	40	785.7	780.6	3.5	19.5	8.1	37.4
Total		540				278				509						

Table 4.7: Performance comparison between LP relaxations of n-ILP$_{\texttt{B}}$, n-ILP$_{\texttt{TF}}$, n-ILP$_{\texttt{B+TF}}$ and c-ILP on `SYNTHETIC` instances

and 0.8. Furthermore, to broaden our analysis, we extend the range of λ values. Specifically, we consider 9 values of λ, ranging from 0.1 to 0.9 with a step size of 0.1. In Figure 4.3, we present the computing time boxplot of c-ILP for 11 groups of 180 instances, where each group has the same number of vertices n. As in Figure 4.2, we graphically represent the computing time using quartiles and we report the number of instances solved to proven optimality (#opt) for each group out of the total number of instances in the group (180). As expected, we observe a direct correlation between the computing time of c-ILP and the number of vertices in the graph. Specifically, for graphs with up to 700 vertices, c-ILP reaches an optimal solution in a reasonable time (less than 300 seconds for the worst case). For graphs with 900 vertices, the average computing time increases with one instance approaching the time limit at 590 seconds. The compact formulation encounters difficulties when dealing with graphs of 1000 vertices, where ten instances remain unsolved. Subsequently, the number of instances solved to proven optimality decreases to 141 out of 180 for $n = 1100$. The most significant disparity is observed for graphs of 1200 vertices, with only 34 instances (slightly less than a quarter of the total) achieving proven optimality. Lastly, for graphs of 1400 vertices, this number decreases to 18 and for larger graphs (1500 vertices), no instances have been solved to proven optimality within the time limit. It is worth noticing that in cases where instances have not been solved to proven optimality, the solver encounters difficulties in establishing good lower and upper bounds.

Figure 4.3: Computing time boxplot of c-ILP on SYNTHETIC instances



4.6 Concluding remarks

In this chapter, an extensive computational analysis was performed to evaluate the performance of the formulations designed for the MFBP. The experiments have shown that the algorithms proposed in this dissertation significantly outperform the direct use of a bilevel solver, representing the current state-of-the-art technique for addressing the MFBP. Moreover, the compact formulation proves to be the most efficient, surpassing the natural formulations and demonstrating its ability to handle large-sized instances efficiently within a reasonable time.

Chapter 5

The multicommodity flow blocker problem : Formulations and polyhedral analysis

Contents

In this chapter, we provide a formal definition of the multicommodity flow blocker problem. We begin in Section 5.1 with a comprehensive overview of the multicommodity flow problem, exploring both its splittable and unsplittable variants. Specifically, we discuss the particular versions of the problem addressed in this thesis, along with graphical illustrations for clarity. Additionally, we introduce mathematical models that have been developed to effectively tackle this problem. Following this, in Section 5.2, we delve into the blocker variant of the multicommodity flow problem, known as the multicommodity flow blocker problem. In this section, we also examine a closely related problem, namely

the multicommodity flow interdiction problem, which has been previously investigated in the literature. Using a reduction to the multicommodity flow interdiction problem, we demonstrate the $\mathcal{NP}-$hardness of the multicommodity flow blocker problem. For the splittable and unsplittable variants of the multicommodity flow problem, graphical illustrations and a complexity analysis are provided. We further explore the relationship between the two variants. We then introduce in Section 5.3 two mathematical formulations for solving the multicommodity flow blocker problem to proven optimality. The first formulation leverages the bilevel nature of the problem, while the second is an Integer Linear Programming (ILP) formulation, characterized by an exponential family of constraints applicable to each variant. This ILP formulation uses only the design variables of the problem and therefore it is solved using a Branch-and-Cut algorithm that will be further described in Chapter 6. Finally, in Section 5.4, a polyhedral analysis of the ILP formulation is conducted, offering deeper insights into its structure and solutions.

5.1 The multicommodity flow problem

This section provides a formal definition of the multi-commodity flow problem, along with mathematical models and a comprehensive example. It serves to clarify the specific variants of the multicommodity flow problem that we will focus on throughout the rest of this study.

As mentioned in Chapter 1, the multi-commodity flow problem is a well-known optimization problem largely studied in the literature. The multicommodity flow problem (MCFP) involves the efficient routing of multiple commodities through a network. In this optimization problem, a directed graph is used to depict the network, with nodes representing routers and arcs representing the connections or links between them. Each commodity is associated with a specific source node, destination node, and a bandwidth corresponding to the quantity of data to be routed.

The objective of the multicommodity flow problem is to determine the optimal flow of each commodity through the network while respecting various constraints. These constraints typically include capacity limits on the arcs, ensuring that the total flow on any arc does not exceed its capacity. Additionally, for each commodity conservation constraints are imposed at each node, excluding the source and the destination, ensuring that the total incoming flow is equal to the total outgoing flow.

In this study, our focus centers on routing a set of commodities with the objective of maximizing a specified profit function. This profit function is formulated by computing the gains obtained from routing one unit of flow from the source to the destination for each commodity, along with a cost function representing transportation costs, communication delays, or other relevant factors.

A variant of the multicommodity flow problem involves routing the entire bandwidth of the commodities through a single path, for each commodity. This specific variant is referred to as the unsplittable multicommodity flow problem (UMCFP). Conversely, when the multicommodity flow allows to split the flow for a commodity into multiple paths, it is said to be splittable. It is worth noticing that in the splittable variant, the

bandwidth corresponds to the maximum quantity of data to be routed, whereas in the unsplittable variant, the entire bandwidth must either be entirely routed, or no flow is routed at all.

In the following sections, we present mathematical formulations to address the multi-commodity flow problem, distinguishing the splittable and unsplittable variants. To this end, we consider a directed graph $G = (V, A)$ with $m = |A|$ arcs and $n = |V|$ vertices. Each arc $a \in A$ has a capacity $c_a \in \mathbb{Z}_+$ and a routing cost (or flow cost) $p_a \in \mathbb{Z}_+$, which corresponds to the cost for routing one unit of flow on arc a. We consider a set of d commodities K where a commodity $k = (s_k, t_k, b_k, \Gamma_k)$ is defined by a source $s_k \in V$, a destination $t_k \in V$, a bandwidth $b_k \in \mathbb{Z}_+^*$ and a reward $\Gamma_k \in \mathbb{Z}_+^*$ corresponding to a reward obtained when sending one unit of flow from s_k to t_k.

5.1.1 An ILP formulation for the MCFP

Let $y_{k,a}$ be a continuous variable between 0 and 1 representing for each commodity $k \in K$, the proportion of bandwidth b_k routed on an arc $a \in A$. A *multi-commodity flow* (MCF) has to respect the following two topologies of constraints. The first set of constraints are called the *capacity constraints*, for the arcs:

$$\sum_{k \in K} b_k \, y_{k,a} \leq c_a, \qquad \forall \, a \in A. \tag{5.1}$$

The capacity constraints ensure that the total flow value on an arc does not exceed its capacity. The second set of constraints is called the *flow-conservation constraints*, for the vertices of the graph:

$$\sum_{a \in \delta^+(u)} b_k \, y_{k,a} - \sum_{a \in \delta^-(u)} b_k \, y_{k,a} = \begin{cases} \lambda_k \text{ if } u = s_k, \\ 0 \text{ if } u \in V \backslash \{s_k, t_k\}, \\ -\lambda_k \text{ if } u = t_k \end{cases} \quad \forall k \in K, \forall u \in V. \tag{5.2}$$

where $\lambda_k \in \mathbb{Z}_+$ is the value of the flow routed for commodity $k \in K$. Note that, by definition, $\lambda_k \leq b_k$. The flow-conservation constraints impose that for every commodity $k \in K$, the flow entering into a vertex that is not the source s_k or the destination t_k is equal to the total flow outgoing that vertex. In addition, they ensure that the flow outgoing the source s_k is equal to the flow entering into the destination t_k. For every commodity $k \in K$, if $\lambda_k = 0$, then commodity k is said to be *unsatisfied*. If $\lambda_k = b_k$, then k is said to be *fully satisfied*. Finally, if $0 < \lambda_k < b_k$, then commodity k is said to be *partially satisfied* and only a portion of the bandwidth is routed from s_k to t_k.

Let $\Omega(G)$ be the function returning the total routing profit of an MCF in a graph G:

$$\Omega(G) = \sum_{k \in K} \Gamma_k \, \lambda_k - \sum_{a \in A} b_k \, p_a \, y_{k,a}.$$

An MCF is characterized by a positive routing profit. Moreover, the maximum profit of a multi-commodity flow, i.e., the maximum value of the function $\Omega(G)$ is denoted by $\Psi(G)$.

In this study, we focus on a particular case of multicommodity flow problems, that asks to determine an MCF with a maximum profit $\Psi(G)$. Therefore a model for the multicommodity flow problem (MCFP) reads as follows:

$$\zeta(\text{MCFP}) = \max_{\mathbf{y}\in[0,1]^{d\times m},\boldsymbol{\lambda}\geq 0}\Big\{\Omega(G) : (5.1), (5.2)\Big\}. \tag{5.3}$$

It is worth noticing that the multicommodity flow problem described in this section is said to be *splittable*. In other words, for each commodity, the flow can be routed across multiple paths. A specific case of the MCFP consists in routing the flow of a commodity through a single path. In this scenario, the MCFP is said to be *unsplittable*. A detailed description of this problem is given in the next section.

5.1.2 An ILP formulation for the UMCFP

Let $y_{k,a}$ be a binary variable associated to a commodity $k \in K$ and an arc $a \in A$ that takes value 1 if and only if the arc a is routing b_k units of flow for commodity k. As the MCFP, an unsplittable multi-commodity flow (UMCF), has to respect the capacity constraints for the arcs, i.e., constraints (5.1), the flow conservation constraints, i.e., constraints (5.2) and a positive profit value. It is worth noticing that in the UMCFP, a commodity can be either fully satisfied of unsatisfied. Accordingly, for every commodity $k \in K$, λ_k can take values 0 or b_k.

As for the MCFP, the UMCFP asks to determine the multi-commodity flow with a maximum profit $\Psi(G)$. Therefore, a model for the UMCFP reads as follows:

$$\zeta(\text{UMCFP}) = \max_{\mathbf{y}\in\{0,1\}^{d\times m},\boldsymbol{\lambda}\geq 0}\Big\{\Omega(G) : (5.1), (5.2)\Big\}. \tag{5.4}$$

It is worth noticing that by definition of a UMCF, it can be defined as a set of paths $\{p_0, \cdots, p_d\}$ such that for every commodity $k \in K$, p_k is a path between s_k and t_k associated to k. If k is satisfied, then the entire bandwidth b_k is routed through the path p_k. Otherwise, if k is unsatisfied, then p_k is represented as an empty set.

Despite the similarities between the MCFP and the UMCFP, where both problems share the same objective function and constraints, the critical distinction between the two problems lies in the allowable range of values for the flow variables $\mathbf{y}$. In the UMCFP, for each commodity $k \in k$ and for every arc $a \in A$, the variables $y_{k,a}$ are binary. In contrast, the MCFP allows continuous values of $y_{k,a}$ within the range $[0, 1]$. Accordingly, while any solution of the UMCFP is also a solution for the MCFP, the reverse is not necessarily true. This distinction is highlighted by the following remarks, which are a direct consequence of the classical relationship between linear programming (LP) and integer linear programming (ILP).

Remark 1. *Any solution of the UMCFP is a solution of the MCFP.*

Remark 2. *Any solution of the MCFP is not necessarily a solution of the MCFP.*

Indeed, let S_{umcf} be a solution of an unsplittable multicommodity flow problem solved in the graph G. For each satisfied commodity, the entire bandwidth is routed through a single path from the source to the destination. It is easy to see that this solution is also a solution for the splittable variant. First, since S_{umcf} is a solution of the UMCFP, it satisfies all constraints of the unsplittable variant, meaning that for each satisfied commodity, the entire bandwidth is routed along a single path. This implies that the solution S_{umcf} satisfies the flow conservation constraints as well as the capacity constraints, which also corresponds to constraints of the splittable variant. Therefore, S_{umcf} is a solution for the MCFP solved in G. However, any solution of the MCFP is not necessarily a solution of the UMCFP. For instance, the splittable variant of the MCFP does not ensure that for every satisfied commodity, the entire bandwidth is routed through a single path.

Remark 3. *Any solution of the MCFP provides an upper bound for the UMCFP.*

In the next section, we give an illustrative example of a multi-commodity flow problem, in its splittable and unsplittable variant.

5.1.3 Graphical illustrations

We illustrate in this section the features of optimal MCFP solutions thanks to an example graph with 8 vertices and 12 arcs shown in Figures 5.1 and 5.2. We consider two commodities $k_1 = (s_1, t_1, b_1 = 8, \Gamma_1 = 6)$ and $k_2 = (s_2, t_2, b_2 = 10, \Gamma_2 = 35)$. We report on each arc two values separated by the symbol ";": the first one, in bold, is the routing cost and the second one, is the capacity of the arc. The curved arcs represent a solution of the MCFP. The red ones are the arcs routing flow for commodity k_1 and the blue ones are the arcs routing flow for commodity k_2. Above these arcs, we report (in red or in blue) the amount of flow routed.

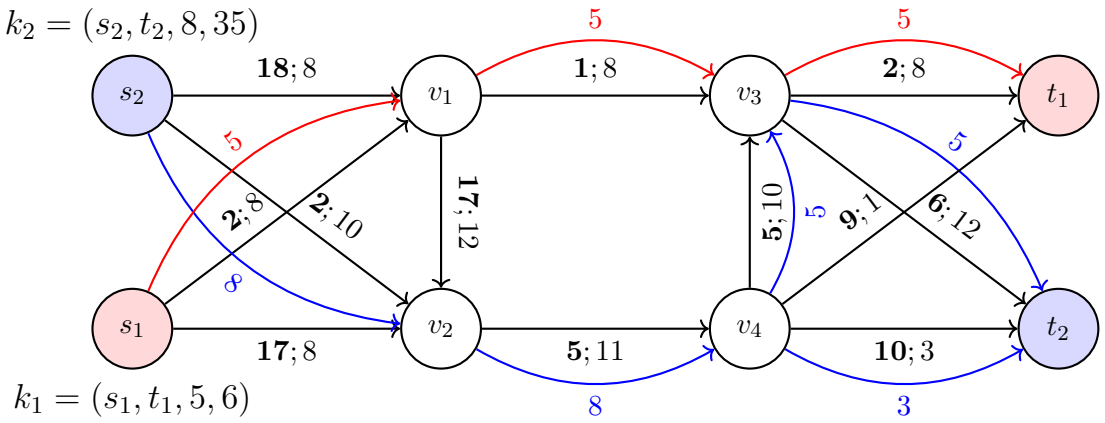


Figure 5.1: An optimal MCFP solution

In Figure 5.1, the two commodities are fully satisfied. The flow for commodity k_1 is routed through a unique path composed by the arcs $(s_1, v_1), (v_1, v_3)$ and (v_3, t_1). The flow for commodity k_2 is split into two paths; 8 units of flow are routed through the arcs (s_2, v_2) and (v_2, v_4), 5 units of flow are routed through the arcs $(v_4, v3)$ and (v_3, t_2)

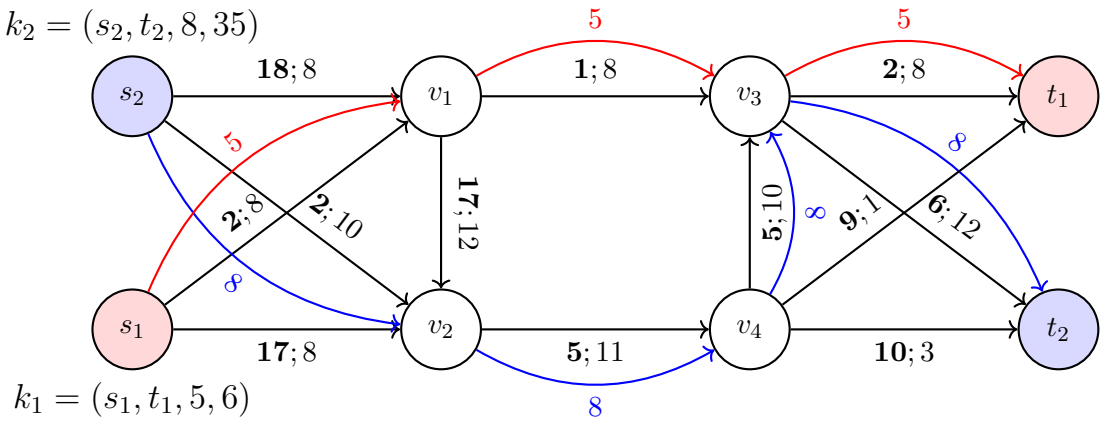


Figure 5.2: An optimal UMCFP solution

and 3 units of flow are routed through the arc (v_4, t_2). The total routing cost is equal to 144.

On the other hand, Figure 5.2 represents a UMCFP solution. Indeed, for each commodity, k_1 and k_2, the bandwidth is routed through a set of arcs composing a path; $\{(s_1, v_1), (v_1, v_3), (v_3, t_1)\}$ for commodity k_1 and $\{(s_2, v_2), (v_2, v_4), (v_4, v_3), (v_3, t_2)\}$ for commodity k_2. As previously, the two commodities are fully satisfied. The total routing profit of the multi-commodity flow represented in this figure is equal to 141.

Remark that these two examples outcome results of Proposition 1 and Corollary 3 since the solution of the UMCFP is equal to 141, which is less than 144, the solution of the MCFP. Moreover, it is easy to see that the solution of the UMCFP is also a solution for the MCFP.

5.2 The multicommodity flow blocker problem

In the previous section, we introduced and described the multi-commodity flow problem. We now focus on the blocker problem applied to the multi-commodity flow problem, which is called the multicommodity flow blocker problem (MCFBP). In what follows, we formally define the problem, emphasizing both the splittable and the unsplittable variants. We recall that in this thesis, we are interested in the Maximum-Profit multi-commodity flow problem, which aims to route traffic through the graph G, with respect to the commodities K while maximizing a total routing profit.

5.2.1 Description of the problem

Given a directed graph $G = (V, A)$ and a set of commodities K, we assign to every arc $a \in A$, a positive integer *blocker cost* $r_a \in \mathbb{Z}_+$, corresponding to the cost for removing the arc a from the graph.

The multicommodity flow blocker problem (MCFBP) consists in finding a minimum-cost subset of arcs to be removed from the graph G, i.e., *blocked*, in such a way that the maximum profit of the MCFP in the remaining graph, also called *non-blocked graph*,

is no larger than a given threshold. The threshold is called the *target profit*, a positive integer value denoted by $\Phi \in \mathbb{Z}_+$. Without loss of generality, we consider instances in which $\Phi < \Omega(G)$, where $\Omega(G)$ is the maximum MCF profit in G, otherwise, clearly, an optimal MCFBP solution is the empty subset of blocked arcs.

We now introduce a variant of the multi-commodity flow blocker problem known as the unsplittable multicommodity flow blocker problem (UMCFBP), wherein the blocker problem is applied to an unsplittable multi-commodity flow problem (UMCFP) rather than a splittable MCFP. Consequently, the distinction between the splittable and unsplittable variants of the MCFBP lies in the definition of the MCFP. As the MCFBP, the UMCFBP aims to identify a minimum-cost subset of blocked arcs in such a way that the maximum profit of the UMCFP in the remaining graph does not exceed the target profit.

Graphical illustrations

We illustrate in this section the features of optimal MCFBP and UMCFBP solutions thanks to the example graph shown in Figures 5.1 and 5.2. The target profit Φ is set to 10. We report on each arc three values. The first one on top is the blocker cost. The two others, on bottom are separated by the symbol ";": the first one, in bold, is the routing cost and the second one, is the capacity of the arc. The blocked arcs are depicted with dashed lines. The curved arcs represent solution of the MCFP in the non-blocked graph. As previously, the red ones are the arcs routing flow for commodity k_1 and the blue ones are the arcs routing flow for commodity k_2. Above these arcs, we report (in red or in blue) the amount of flow routed.

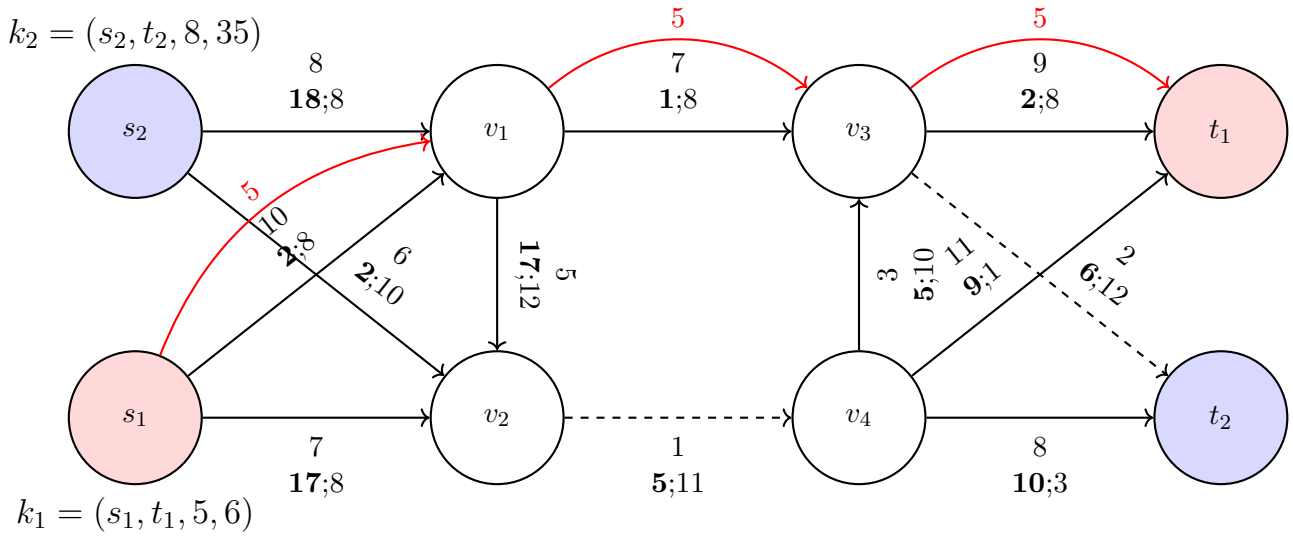


Figure 5.3: An optimal MCFBP solution

Figure 5.3 represents an optimal MCFBP solution with 2 blocked arcs, (v_2, v_4) and (v_3, t_2), having a total blocker cost of 3. Only one commodity, k_1, is fully satisfied in the remaining graph. The second commodity k_2 is not satisfied. The total routing profit

of the multi-commodity flow remaining in the non-blocked graph after the removal of these two arcs is equal to 5, which is less than the target profit Φ set to 30.

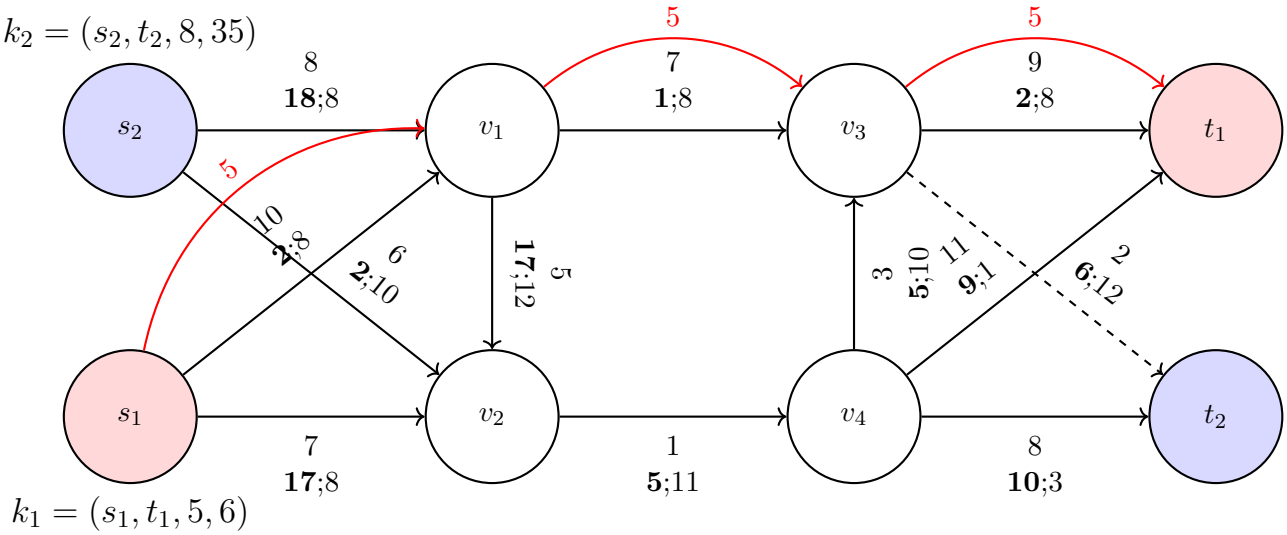


Figure 5.4: An optimal UMCFBP solution

On the other hand, Figure 5.4 represents a UMCFBP solution for the same target profit value, i.e., $\Phi = 10$. Only one arc is blocked, the arc (v_3, t_2) with a total blocker cost equal to 2. The MCF remaining in the non-blocked graph is a UMCF since the complete bandwidth of commodity k_1 is routed through a single path. Note that, as previously, commodity k_2 is unsatisfied. The total profit is equal to $5 < 10$.

It is worth noticing that for the MCFBP, two arcs need to be blocked. Specifically, if the arc (v_2, t_1) remains in the network, another MCFP with a profit exceeding 10 persists in the network. In contrast, a UMCFP with a profit greater than 10 cannot remain in the same graph.

5.2.2 Complexity

This section delves into the complexity of the multicommodity flow blocker, examining both its splittable and unsplittable variants. Beginning with the unsplittable variant, the next proposition investigates the decision problem of the UMCFBP following an analysis of its optimization problem. At the end on the section, our focus turns to the splittable variant of the MCFBP.

Proposition 13. *The decision problem of the UMCFBP is not in $\mathcal{NP}$.*

Proof. Given a subset of blocked vertices and a target profit value Φ, the task of determining whether the remaining graph, i.e., the graph remaining after removal of the blocked arcs does not support a UMCF with a total profit exceeding Φ requires solving the decision problem of the UMCFP, which is known to be $\mathcal{NP}-$complete (see Even et al. [1975]).

□

We now introduce the *most vital edges for the shortest path problem* (MVESPP). Let $G = (V, A)$ be a directed graph with two distinct vertices $s \in V$ and $t \in V$. Each arc $a \in A$ is associated with an edge-length $l_a \in \mathbb{Z}_+$. The MVESPP asks to determine a minimum set of edges to remove from the graph in such a way that there does not exist a path between s and t whose total edge length is less than or equal to L. This problem is $\mathcal{NP}$-hard, as shown in Khachiyan et al. [2008].

By reducing the MVESPP to the UMCFBP, the next proposition characterizes the computational complexity of the latter one.

Proposition 14. *The UMCFBP is $\mathcal{NP}$-hard.*

Proof. Starting from a MVESPP instance, we set $r_a = l_a$, $c_a = p_a = 1$, for every arc $a \in A$. We consider a single commodity $k_0 = (s, t, b_0, \Gamma_0)$, with $b_0 = 1$ and $\Gamma_0 = M$, where M is a large constant value. The target flow value Φ is set to value L. Once the UMCFBP is solved, its optimal solution corresponds to an optimal MVESPP solution. Indeed, solving a UMCFP in the non-blocked graph leads to finding a path between s and t, since all arcs have unitary capacities. In addition, given the large reward Γ_0 associated with the single commodity k_0, maximizing the overall routing profit leads to minimizing the total routing cost along the chosen path. Finally, as all blocker costs are unitary, finding the minimum-cost subset of blocked arcs leads to finding the most vital edges. Therefore, the optimal solution of the UMCFBP corresponds to an optimal solution of the MVESPP. This shows that the UMCFBP is $\mathcal{NP}$-hard. □

In the remainder of this section, our focus is directed toward the investigation of the computational complexity of the UMCFBP in instances where the target profit value is set to zero, implying that no UMCF exists in the non-blocked graph.

For this purpose, we introduce the *minimum multicut problem* (MMP). Given a directed graph $G = (V, A)$ where all arcs $a \in A$ have a positive weight ω_a, and a set of d source-destination pairs of vertices $\{(s_i, t_i), i \in [0, d-1], s_i, t_i \in V\}$, the MMP aims to find a minimum weight set of arcs $A' \subseteq A$ such that the removal of A' disconnects each pair. The MMP is $\mathcal{NP}$-hard for $d \geq 3$, as proved in Dahlhaus et al. [1992]. By reducing the MMP to the MCFBP, the next proposition characterizes the computational complexity of the latter one.

Proposition 15. *For $|K| \geq 3$ the UMCFBP is $\mathcal{NP}$-hard even if $\Phi = 0$.*

Proof. Starting from an instance of the MMP, we set $p_a = \omega_a$ and we define a set of commodities K as a set of d source-destination pairs of vertices $\{(s_i, t_i), i \in [0, d-1], s_i, t_i \in V\}$ with a bandwidth $b_k = 1$ and a reward $\Gamma_k = M$, where M is a large constant value, for each $k \in K$. By setting the target profit Φ to zero, it follows that no flow can be routed in the non-blocked graph. Accordingly, once the UMCFBP is solved, its optimal solution corresponds to the minimum weight set of arcs $A' \subseteq A$ for which there is no flow of value at least 1 between any pair of sources and destinations (s_k, t_k) for all commodities $k \in K$. In other words, the set of arcs A' disconnects each pair of vertices and therefore it is an optimal solution of the MMP.

□

It is worth noticing that, given an MCFBP instance, if $b_k = 1$ for a commodity $k \in K$, then this commodity can be either routed through a unique path or unsatisfied. Accordingly, if $b_k = 1$ for all commodities $k \in K$, the follower problem of the MCFBP is a UMCFP and therefore solving the MCFBP reduces to solving the UMCFBP. Accordingly, the next proposition characterizes complexity of the MCFBP.

Proposition 16. *The MCFBP is $\mathcal{NP}$-hard, even if $\Phi = 0$.*

Proof. Starting from an instance of the UMCFBP, we set $b_k = 1$ for all commodities $k \in K$. Once the MCFBP solution is solved, its optimal solution corresponds to an optimal UMCFBP solution. Which shows that the MCFBP is $\mathcal{NP}$-hard.

□

Proposition 16 demonstrates the $\mathcal{NP}$-hardness of the MCFBP thanks to a reduction from the UMCFBP. In the next section, we delve deeper to establish a connection between solutions of the MCFBP and solutions of the UMCFBP.

5.2.3 Relation between the MCFBP and the UMCFBP

As mentioned earlier, the distinction between the MCFBP and the UMCFBP lies in the definition of the multicommodity flow problem. As shown in Section 5.1, the splittable and unsplittable variants of the MCFP exhibit a close relationship. Consequently, the corresponding blocker problems for both variants also share a strong connection. More precisely, a relationship exists that links solutions of both problems, as demonstrated in this section.

The next proposition and corollary exhibit a relationship between solutions of the MCFBP and solutions of the UMCFBP.

Indeed, based on Remark 1 and Remark 3, it can be established that all solutions of the MCFBP are feasible solutions for the UMCFBP. However, the converse is not necessarily true. This observation leads to the following proposition.

Proposition 17. *Any solution of the MCFBP is a solution of the UMCFBP.*

Proof. Let S be a solution for the blocker variant of an MCFP solved in G. Since any solution of the UMCFP is a solution for the MCFP (see Proposition 1), S is also a solution for the blocker variant of a UMCFP solved in G. Therefore, any solution of the MCFBP is a solution for the UMCFBP.

□

It is worth noticing that the reverse of Proposition 17 does not hold, i.e., any solution of the UMCFBP is not necessarily a solution of the MCFBP. Indeed, the blocker variant of the UMCFP ensures that it does not exist a UMCF with a profit greater than Φ in the remaining graph, i.e., the graph remaining after the removal of the blocked arcs. However, it does not guarantee the absence of a splittable MCF with a profit greater than Φ, since any solution of the MCFP is not necessarily a solution of the UMCFP.

According to Proposition 17, the next corollary establishes a relationship between solution values of the MCFBP and the UMCFBP.

Corollary 4. *Solution of the MCFBP provides an upper bound for the UMCFBP.*

5.3 Formulations

In this section, we first introduce a bilevel formulation designed for the MCFBP and the UMCFBP. We then present an ILP formulation to solve both variants (Bentoumi et al. [2023c]).

5.3.1 A bilevel formulation for the multicommodity flow blocker problem

We recall that the continuous vector $\mathbf{y} \in [0,1]^{d \times m}$ is the flow vector associated with an MCF.

As for the maximum flow blocker problem (MFBP) studied in Chapter 3, we introduce a vector $\mathbf{x}$, called *blocker vector*, of m binary decision variables, each variable $x_a \in \{0,1\}$ is associated to an arc $a \in A$ and it takes value 1 if and only if the arc a is blocked. We denote by $\mathcal{A}(\mathbf{x})$ the set of blocked arcs and $\mathcal{G}_{NB}(\mathbf{x})$, the *non-blocked graph* which is the graph remaining after removal of the blocked arcs, i.e. the graph $\mathcal{G}_{NB}(\mathbf{x}) = (V, A \backslash \mathcal{A}(\mathbf{x}))$.

The MCFBP aims to find a set of blocked arcs $\mathcal{A}(\mathbf{x})$, with a minimum total blocker cost and such that the total profit of the MCF remaining in the non-blocked graph $\mathcal{G}_{NB}(\mathbf{x})$ is no larger than Φ, i.e., $\Psi(\mathcal{G}_{NB}(\mathbf{x})) \leq \Phi$. In this bi-level problem, there are two types of variables. The first-level decision variables, corresponding to the blocker vector $\mathbf{x}$, are associated with the so-called *leader problem*. These variables affect the second-level decision variables, corresponding to the flow variables $\mathbf{y}$ and associated with the so-called *follower problem*. The leader determines a set of blocked arcs to be removed from the graph and the follower determines the maximum profit of the MCF remaining in the non-blocked graph. The leader anticipates the optimal follower's solution with the goal of choosing the minimum cost subset of arcs to be blocked that results in a remaining graph having an MCF with a maximum profit no larger than Φ. Accordingly, a bi-level formulation for the MCFBP reads as follows:

$$\min_{\mathbf{x} \in \{0,1\}^m} \sum_{a \in A} r_a \, x_a \tag{5.5a}$$

$$\vartheta(\mathbf{x}) \leq \Phi, \tag{5.5b}$$

$$\text{where} \quad \vartheta(\mathbf{x}) = \max_{\mathbf{y} \in [0,1]^{d \times m}} \left\{ \Psi(G) : (5.2), \sum_{k \in K} b_k y_{k,a} \leq c_a(1 - x_a), \ \forall \, a \in A. \right\} \tag{5.5c}$$

The link between the leader problem ((5.5a)-(5.5b)) and the follower problem given by (5.5c) is established by the *value function* $\vartheta(\mathbf{x})$ which returns the maximum routing profit in the non-blocked graph. This value is determined by the follower objective function $\Psi(G)$ which corresponds to the maximum routing profit in the graph where the capacity of blocked arcs is set to 0. Constraint $\sum_{k \in K} b_k \ y_{k,a} \leq c_a(1 - x_a)$ models the capacity constraint of the arc $a \in A$, see (5.1), and impose, at the same time, a flow of value 0 on blocked arcs. As far as the leader problem is concerned, the objective function (5.5a) minimizes the total blocker cost and constraint (5.5b) imposes that the maximum routing profit $\vartheta(\mathbf{w})$ is no larger than Φ.

We define a *feasible MCFBP solution* as a set of arcs $\mathcal{A}(\mathbf{x}) \subseteq A$ whose removal results in no MCF with a profit greater than Φ existing in the non-blocked $\mathcal{G}_{NB}(\mathbf{x})$.

A first single-level reformulation for the MCFBP

In this section, we present a first reformulation of the bilevel model (5.5) for the MCFBP into a single-level model, using a "dualize-and-combine" method (see Wei and Walteros [2022]).

We recall that given any blocker solution $\hat{\mathbf{x}}$, the follower problem is a multi-commodity flow problem. Therefore, for fixed $\hat{\mathbf{x}}$, the dual model of the follower problem (5.5c) reads as follows:

$$\min \sum_{a \in A} \Theta_a \, c_a \, (1 - \hat{x}_a) \tag{5.6a}$$

$$\pi^k_{t_k} - \pi^k_{s_k} \leq \Gamma_k \qquad \forall k \in K, \tag{5.6b}$$

$$\pi^k_v - \pi^k_u - \Theta_a \leq p_{(u,v)} \qquad \forall k \in K, \forall (u,v) \in A, \tag{5.6c}$$

$$\pi^k_u \text{ unrestricted} \qquad \forall k \in K, \forall u \in V, \tag{5.6d}$$

$$\Theta_a \geq 0, \qquad \forall a \in A. \tag{5.6e}$$

where π^k_u is the dual variable associated with Equality (5.2) and Θ_a is the dual variable associated with constraint "$\sum_{k \in K} y_{k,a} b_k \leq c_a \, (1 - x_a)$". By using this reformulation of the follower problem in the bilevel Model (5.5), we obtain the following single-level formulation to solve the MCFBP.

$$\min_{\mathbf{x} \in \{0,1\}^m} \sum_{a \in A} r_a \, x_a \tag{5.7a}$$

$$\sum_{a \in A} \Theta_a \, c_a \, (1 - x_a) \leq \Phi \tag{5.7b}$$

$$\pi^k_{t_k} - \pi^k_{s_k} \leq \Gamma_k \qquad \forall k \in K, \tag{5.7c}$$

$$\pi^k_v - \pi^k_u - \Theta_a \leq p_{(u,v)} \qquad \forall k \in K, \forall (u,v) \in A, \tag{5.7d}$$

$$\pi^k_u \text{ unrestricted} \qquad \forall k \in K, \forall u \in V, \tag{5.7e}$$

$$\Theta_a \geq 0, \qquad \forall a \in A. \tag{5.7f}$$

It is worth noticing that Model (5.7) is a mixed-integer bilinear programming model that can be addressed using standard linearization techniques. However, a similar approach was developed in Lim and Smith [2007a] for the multi-commodity flow interdiction problem, but it proved to be inefficient in practice. Hence, we introduce an alternative single-level reformulation of the bilevel problem in the next section.

5.3.2 A second single-level ILP formulation for the MCFBP

In this section, we introduce a second reformulation of the bilevel model (5.5) for the MCFBP, using a penalty formulation. In other words, for a given blocker policy $\hat{x}$, for each arc $a \in A$ and for each commodity $k \in K$, let M_a^k be a large constant value that serves as a penalization term in the objective function of the follower problem. This penalty represents the cost for routing one unit of flow for commodity k through a blocked arc a, i.e., an arc a for which $\hat{x}_a = 1$. Therefore, the follower problem can be reformulated as follows:

$$\vartheta(\mathbf{x}) = \max_{\mathbf{y} \in [0,1]^{d \times m}} \left\{ \Psi(G) - \sum_{k \in K} \sum_{a \in A} M_a^k \, \hat{x}_a \, y_{k,a} : (5.2), (5.1) \right\}. \tag{5.8}$$

In this reformulation, constraints "$\sum_{k \in K} y_{k,a} b_k \leq c_a \ (1 - x_a)$" of the follower problem are replaced with the "standard" capacity constraints (5.1) for the arcs. The constraints of the follower do not depend anymore on the first-level variables and a penalization term is added to the new objective function.

Two approaches could be explored based on the penalty formulation (5.8) of the follower problem. One approach involves investigating the dual of the follower problem as presented in Model (5.8). Alternatively, given that the feasible region of the follower problem remains unchanged for various blocker values $\mathbf{x}$, another approach consists in applying a Benders cutting plane algorithm in which the master program is a mixed-integer programming problem with binary blocker variables and additional continuous variable coming from the follower problem that is a multicommodity flow problem.

In what follows our focus will be on the Benders cut approach. Consequently, the objective is to address the mixed-integer master problem by incorporating Benders cuts at each iteration.

To this end, let us introduce the polytope $\mathcal{P}_{mcf}$ of feasible solutions for the follower subproblem which does not depend on the leader variables as shown in Model (5.8):

$$\mathcal{P}_{mcf} = \left\{ \boldsymbol{y} \in [0,1]^m : \Omega(\mathbf{y}) \geq \Phi + 1, \ \ (5.1), \ (5.2). \right\}, \tag{5.9}$$

where $\Omega(\mathbf{y})$ returns the total routing profit of the MCF defined by vector $\mathbf{y}$.

We remark that only multicommodity flows of value strictly larger than Φ are associated with Constraint (5.5b) of the bilevel model for the MCFBP, i.e, Model (5.5). This can be imposed by a "$\geq \Phi + 1$" constraint since all the capacities of the arcs are integer values. The model (5.8) is valid for any (fractional) vector $\boldsymbol{x} \in [0,1]^m$ and, since the objective function is linear, it is sufficient to optimize over the set of extreme points $\boldsymbol{y}$ of $\mathcal{P}_{mcf}$ (denoted $ext(\mathcal{P}_{mcf})$). The constraints (5.5b) can then be restated as follows:

$$\vartheta(\boldsymbol{x}) = \max_{\boldsymbol{y} \in ext(\mathcal{P}_f)} \left\{ \Omega(\mathbf{y}) - \sum_{k \in K} \sum_{a \in A} M_a^k x_a y_{k,a} \right\} \leq \Phi. \tag{5.10}$$

Accordingly, by applying a Benders-like decomposition to the bilevel model (5.5), we

obtain the following single-level ILP formulation for the MCFBP:

$$\min_{\boldsymbol{x} \in \{0,1\}^m} \quad \sum_{a \in A} r_a \, x_a \tag{5.11a}$$

$$\sum_{k \in K} \Gamma_k \, \lambda_k - \sum_{a \in A} b_k \, p_a \, y_{k,a} - \sum_{k \in K} \sum_{a \in A} M_a^k \, x_a \, y_{k,a} \leq \Phi, \quad \forall \, \boldsymbol{y} \in ext(\mathcal{P}_{mcf}). \tag{5.11b}$$

where constraints (5.5b) are replaced with constraints (5.11b), called *Benders cuts*, an exponential-size family of constraints, one for each extreme point of $\mathcal{P}_{mcf}$.

Some studies, such as those detailed in Lim and Smith [2007a], have been conducted to determine an appropriate value for the constant M_a^k in the purpose of solving the multi-commodity flow interdiction problem. For the MCFBP presented in this thesis, one straightforward approach involves setting M_a^k equal to $p_a \times b_k$. Selecting the appropriate value for M_a^k is crucial, as it directly influences the formulation and the efficiency of the model. Consequently, in the following section, we introduce a generalization of the Benders cut inequalities (5.11b) that do not require the determination of such a constant.

It is worth noticing that the bilevel formulation (5.5) remains valid for the UMCFBP. The distinction lies in the definition of the follower problem which is a UMCFP instead of an MCFP. In this scenario, in accordance with the models of both the MCFP and the UMCFP, the only difference lies in the definition set of variables $\mathbf{y}$. However, solving the bilevel model for the UMCFBP poses challenges, primarily due to the binary nature of the variables $\mathbf{y}$, which makes the dualization of the follower problem not possible. Instead, alternative techniques stemming from bilevel optimization can be employed. Specifically, within the realm of bilevel optimization, a category known as *generalized bilevel programming*(see Kleinert et al. [2021]) offers increased flexibility and generality for modeling specific problems. The Benders cut approach remains applicable for the UMCFBP when considering the polytope $\mathcal{P}_{umcf}$ of feasible solutions for the follower problem, which is defined as:

$$\mathcal{P}_{umcf} = \left\{ \boldsymbol{y} \in \{0,1\}^m : \Omega(\mathbf{y}) \geq \Phi + 1, \;\; (5.1), \; (5.2). \right\}, \tag{5.12}$$

In the following section, we present an ILP formulation to tackle both the MCFBP and the UMCFBP.

5.3.3 An ILP formulation for the multicommodity flow blocker problem

In this section, we present a model to solve the MCFBP using an exponential number of constraints in the natural space of the blocker variables, introduced in Section 5.3.1. This formulation employs a set-covering approach, providing a clear and concise structure that can be easily adjusted to tackle different variants of the problem.

Let $G = (V, A)$ be a directed graph and K be a set of commodities in G. For a given vector $\boldsymbol{y} \in ext(\mathcal{P}_{mcf})$, we define the subset of arcs $\mathcal{A}_S(\boldsymbol{y}) \subseteq A$ routing a multicommodity flow in the extreme point $\boldsymbol{y}$ as follows:

$$\mathcal{A}_S(\boldsymbol{y}) = \Big\{a \in A : y_{k,a} > 0, \forall k \in K\Big\}.$$

It is worth noticing that according to the definition of $\mathcal{P}_{mcf}$, $\mathbf{y}$ respects all the constraints associated with the MCFP. Moreover, these arcs induce the *support graph* $\mathcal{G}_S(\boldsymbol{y}) = (V, \mathcal{A}_S(\boldsymbol{y}))$ in which, by construction, the maximum MCFP profit $\Psi(\mathcal{G}_S(\boldsymbol{y}))$ is larger than or equal to $\Phi + 1$.

We recall that $\mathcal{A}(\mathbf{x})$ is the set of blocked arcs induced by a binary realization of the blocker vector $\mathbf{x}$. In other words, $\mathcal{A}(\mathbf{x})$ is the set of arcs $a \in A$ with x_a equal to 1. Suppose that $\mathcal{A}(\mathbf{x})$ and $\mathcal{A}_S(\boldsymbol{y})$ satisfies the following inequality:

$$|\mathcal{A}(\mathbf{x}) \cap \mathcal{A}_S(\boldsymbol{y})| \geq 1, \quad \boldsymbol{y} \in ext(\mathcal{P}_{mcf}). \tag{5.13}$$

If so, then $\mathbf{x}$ is a feasible solution for the MCFBP. Indeed, the inequality (5.13) guarantees that no MCF with a profit greater than Φ remains in the non-blocked graph. Precisely, the existence of such an MCF would entail the removal of an arc, resulting in an obstruction of the routing process.

Based on this reasoning, a valid ILP formulation for the MCFBP reads as follows:

$$\min \quad \sum_{a \in A} r_a x_a \tag{5.14a}$$

$$\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1, \qquad \boldsymbol{y} \in ext(\mathcal{P}_{mcf}), \tag{5.14b}$$

$$x_a \in \{0, 1\} \qquad a \in A. \tag{5.14c}$$

The objective function (5.14a) minimizes the total blocker cost. Constraints (5.14b), denoted as *target profit* inequalities, are derived from Constraints (5.13). These constraints ensure that at least one arc from every MCF with a total profit greater than Φ, is blocked.

The target profit inequalities (5.14b) are in exponential number. Accordingly, in order to solve the natural formulation (5.14), one needs to implement a Branch-and-Cut (B&C) algorithm where target profit inequalities are separated in the nodes of the branching tree for integer and fractional solutions. This exact algorithm requires defining a *relaxed master problem* (RMP) where the binary variables are replaced with continuous variables taking values between 0 and 1. In the initialization phase only a subset of constraints are included in the RMP. To check that RMP solutions respect all the target profit

inequalities or to determine one or more violated constraints which are then added to the RMP, we propose several separation procedures that will be described in the next chapter.

It is worth noticing that Model (5.14b) is valid for the MCFBP, as well as the UMCFBP. The difference comes from the definition of $\mathcal{P}_{mcf}$. For the UMCFBP, we consider unsplittable flows, i.e., $\mathcal{P}_{mcf}$ should be replaced by the polytope $\mathcal{P}_{umcf}$ of feasible solutions for the UMCFP, defined in 5.12.

Accordingly, for the UMCFBP, the target profit inequalities (5.14b) are replaced by the following inequalities, called *u-target profit inequalities*:

$$\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1, \qquad \boldsymbol{y} \in ext(\mathcal{P}_{mcf}). \tag{5.15}$$

A valid ILP formulation for the UMCFBP reads as follows:

$$\min \quad \sum_{a \in A} r_a x_a \tag{5.16a}$$

$$\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1, \qquad \boldsymbol{y} \in ext(\mathcal{P}_{umcf}) \tag{5.16b}$$

$$x_a \in \{0,1\} \qquad a \in A. \tag{5.16c}$$

As mentioned previously, (see Remark 4), every solution of the UMCFP is a valid solution for the MCFP. Accordingly, we have the following relation between the two polytopes $\mathcal{P}_{mcf}$ and $\mathcal{P}_{umcf}$:

$$\mathcal{P}_{umcf} \subseteq \mathcal{P}_{mcf}.$$

This shows that inequalities (5.15) are valid inequalities for the natural formulation (5.14). Precisely, inequalities (5.15) are contained in the set of inequalities (5.14b).

In the next section, we propose a polyhedral analysis of Model (5.14) and Model (5.16).

5.4 Polyhedral analysis

This section develops a polyhedral analysis for the natural formulation of the MCFBP given by Model (5.14) with target profit inequalities (5.14b) and the natural formulation for the UMCFBP given by model (5.16) with u-target profit inequalities (5.16b) .

We refer the reader to Chapter 1 for an in-depth discussion of polyhedral analysis concepts.

5.4.1 Associated polytopes

Given a directed graph $G = (V, A)$ and a set of commodities K, we consider the following hypothesis for the rest of this study.

Hypothesis 1. *There exists no arc (s_k, t_k) for all commodities $k \in K$.*

To ensure that Hypothesis 1 is consistently satisfied, a preprocessing operation on the graph can be performed. The procedure consists of examining for each commodity $k \in K$ if there is an arc (s_k, t_k). If such an arc exists, an intermediate vertex v_k can be created, and the arc (s_k, t_k) is replaced by two new arcs, (s_k, v_k) and (v_k, t_k). These two arcs should have the same capacity as the original arc (s_k, t_k), i.e., $c_{(s_k,v_k)} = c_{(v_k,t_k)} = c_{(s_k,t_k)}$. Moreover, we set $p_{(s_k,v_k)} = p_{(s_k,t_k)}$ and $p_{(v_k,t_k)} = 0$.

Let $S \subseteq A$ be a set of blocked arcs. For each solution S of an MCFBP or a UMCFBP, we denote the corresponding *incidence vector* as $\mathbf{x}^S$ where $\mathbf{x}^S \in \{0,1\}^m$ is defined by:

$$x_a^S = \begin{cases} 1 & \text{if } a \in S, \\ 0 & \text{otherwise.} \end{cases}$$

We now introduce the two polytopes of the MCFBP and the UMCFBP, defined respectively by the target profit inequalities (5.14b) and the u-target profit inequalities (5.16b).

Let $P(G, K)$ be the convex hull of the solutions of Formulation (5.14), that is

$$P(G, K) = conv(\{\mathbf{x} \in \{0,1\}^m \quad : \quad \mathbf{x} \text{ satisfies (5.14b)}\}. \tag{5.17}$$

Let $P_U(G, K)$ be the convex hull of the solutions of formulation (5.16), that is

$$P_U(G, K) = conv(\{\mathbf{x} \in \{0,1\}^m \quad : \quad \mathbf{x} \text{ satisfies (5.16b)}\}. \tag{5.18}$$

In the remainder of this section, we will discuss $P(G, K)$ and $P_U(G, K)$ by giving their dimension and describing necessary and sufficient conditions for inequalities of formulations (5.14) and (5.16) to be facet defining.

It is worth noticing that the proofs for both the splittable and unsplittable variants exhibit notable similarities. Specifically, the splittable variant of the MCFBP is a generalization of the unsplittable variant. In the splittable variant, each satisfied commodity allows for the flow to be routed across multiple paths instead of being restricted to a single path. To this end, we primarily focus on developing the proof for the unsplittable variant. We then expand our results to the splittable variant.

Proposition 18. *$P_U(G, K)$ is full dimensional.*

Proof. We need to exhibit $m+1$ solutions such that their incidence vectors are affinely independent. Let $S_0 = A$. Clearly, S_0 is a feasible UMCFBP solution. For every arc $a \in A$, let $S_a = A \setminus \{a\}$. As stated previously by Hypothesis 1, given a commodity k, the source s_k and the destination t_k are linked by at least two arcs. Therefore, even when focusing on a single commodity, i.e., $d = 1$, keeping any one arc in the network is a feasible UMCFBP solution. This corresponds to the set of m solutions $\{S_a, a \in A\}$. Finally, $\{S_a, a \in A\}$ and S_0 constitute a set of $m+1$ solutions for which their incidence vectors are affinely independent.

□

Proposition 19. *$P(G, K)$ is full dimensional.*

Proof. As for $P_U(G, K)$, to prove that $P(G, K)$ is full dimensional, we need to exhibit $m + 1$ solutions such that their incidence vectors are affinely independent. We remark that the two sets of solutions S_0 and $\{S_a, a \in A\}$ described for $P_U(G, K)$ are also feasible MCFBP solutions. Moreover, their incidence vectors are affinely independent. Which ends the proof.

□

5.4.2 Trivial inequalities

This section focuses on the trivial inequalities, i.e., $x_a \geq 0$ and $x_a \leq 1$ for every arc $a \in A$. Note that these trivial inequalities are common to the two natural formulations, i.e., Model (5.14) for the MCFBP and Model (5.16) for the UMCFBP. More precisely, we prove that the trivial inequalities are facets of the polytopes $P(G, K)$ and $P_U(G, K)$.

Proposition 20. *For an arc $a \in A$, inequality $x_a \leq 1$ defines a facet of $P_U(G, K)$.*

Proof. We need to exhibit m feasible solutions for which arc a is blocked i.e. $x_a = 1$, and such that their incidence vectors are affinely independent. Let $S_0 = A$ be a feasible UMCFBP solution since when removing all arcs from the graph, no UMCF can be routed. For every arc $a' \in A \setminus \{a\}$, let $S_{a'} = A \setminus \{a'\}$. Clearly, these sets of arcs are feasible UMCFBP solutions due to the preprocessing performed. More precisely, since two nodes (s_k, t_k), for every commodity $k \in K$, cannot be linked by a unique arc due to Hypothesis 1, no UMCF can be routed in a graph having only one arc. Moreover, the incidence vectors of S_0 and $\{S_{a'}, a' \in A \setminus a\}$ are affinely independent. Which ends the proof. □

Proposition 21. *For an arc $a \in A$, inequality $x_a \leq 1$ defines a facet of $P(G, K)$.*

Proof. As for $P_U(G, K)$, we need to exhibit m feasible solutions for which arc a is blocked, i.e., $x_a = 1$ and such that their incidence vectors are affinely independent. Clearly, the two sets of solutions S_0 and $\{S_{a'}, a' \in A \setminus a\}$ described for $P_U(G, K)$ are also feasible MCFBP solutions. Moreover, their incidence vectors are affinely independent. Which ends the proof. □

Proposition 22. *For an arc $a \in A$, the inequality $x_a \geq 0$ defines a facet of $P_U(G, K)$ if and only if it does not exist a UMCF with a profit greater than Φ that satisfies a subset of commodities $K' \subseteq K$, where $|K| \geq |K'| \geq 1$, all having the same source $s \in V$ and the same destination $t \in V$, connected through a path $\{s, u, t\}$, $u \in V$.*

Proof. ($\Rightarrow$) Suppose that all commodities have the same source and the same destination, linked by two arcs with a sufficient capacity, i.e., such that the capacity of each arc is at least equal to the total bandwidths of all commodities. Moreover, assuming that these two arcs form a UMCF with a profit greater than the target profit value Φ. Then, we have the following valid inequalities:

$$x_{(s,u)} + x_{(u,t)} \geq 1 \tag{5.19}$$

$$x_{(s,u)} \leq 1 \tag{5.20}$$

$$x_{(u,t)} \geq 0 \tag{5.21}$$

Since Inequality (5.21) can be obtained as a linear combination of Inequality (5.19) and Inequality (5.20), it cannot define a facet.

($\Leftarrow$) Let $S_a = A \backslash \{a\}$ and $S_{a,a'} = S_a \backslash \{a'\}$ for all arcs $a \in A, a' \in A$. Due to the condition of the proposition, it cannot exist a UMCF with a profit greater than Φ composed by a single path having two arcs (s,u) and (u,t). Therefore, the sets $\{S_a, a \in A\}$ and $\{S_{a,a'}, a \in A, a' \in A \setminus a\}$ are UMCFBP solutions since it cannot exist a UMCF with a profit greater than Φ with only two arcs remaining in the network. Moreover, the incidence vectors of these two sets are affinely independent.

□

Proposition 23. *For an arc $a \in A$, the inequality $x_a \geq 0$ defines a facet of $P(G,K)$ if and only if it does not exist an MCF with a profit greater than Φ that satisfies a subset of commodities $K' \subseteq K$, where $|K| \geq |K'| \geq 1$, all having the same source $s \in V$ and the same destination $t \in V$, connected through a path $\{s,u,t\}$, $u \in V$.*

Proof. We can derive the proof established for Proposition 22 for the splittable variant of the MCF. Indeed, due to the condition of the proposition, it cannot exist an MCF with a profit greater than Φ with only two arcs remaining in the network. This shows, as previously, that if the condition is respected, the two sets described earlier, $\{S_a, a \in A\}$ and $\{S_{a,a'}, a \in A, a' \in A \setminus a\}$ are MCFBP solutions with affinely independent vectors. Which ends the proof.

□

5.4.3 target profit inequalities

In this section, we focus on the u-target profit inequalities 5.16b for the UMCFBP and the target profit inequalities (5.14b) for the MCFBP.

Proposition 24. *For every $\boldsymbol{y} \in ext(\mathcal{P}_{umcf})$, inequality $\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1$ defines a facet of $P_U(G,K)$ if and only if the UMCF induced by vector $\boldsymbol{y}$ is minimal, i.e., for each arc $\bar{a} \in \mathcal{A}_S(\boldsymbol{y})$, the support graph $\bar{\mathcal{G}}_S(\boldsymbol{y}) = (V, \mathcal{A}_S(\boldsymbol{y}) \setminus \{\bar{a}\})$ does not contain a UMCF with a profit greater than Φ.*

Proof. ($\Rightarrow$) Suppose there is an arc $\bar{a}$ such that the support graph $\bar{\mathcal{G}}_S(\boldsymbol{y}) = (V, \mathcal{A}_S(\boldsymbol{y}) \setminus \{\bar{a}\})$ contains a UMCF with a profit greater than Φ. In this case, we can derive the following inequalities: $\sum_{a \in \mathcal{A}_S(\boldsymbol{y}) \setminus \{\bar{a}\}} x_a \geq 1$ and $x_{\bar{a}} \geq 0$. Accordingly, the inequality $\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1$ cannot define a facet of $P_U(G,K)$ since it can be obtained as a linear combination of these inequalities.
($\Leftarrow$) We denote by $gx \geq \alpha$ the inequality $\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1$. Let $bx \geq \beta$ be an inequality that defines a facet of $P_U(G,K)$. Suppose that $\{\mathbf{x} \in P : g\mathbf{x} = \alpha\} \subseteq \{\mathbf{x} \in P : b\mathbf{x} = \beta\}$.

We will show that there exists a scalar $\rho > 0$ such that $b = \rho g$. For $\bar{a} \in \mathcal{A}_S(\boldsymbol{y})$ and $\bar{\bar{a}} \in \mathcal{A}_S(\boldsymbol{y}) \setminus \{\bar{a}\}$, let $S_{\bar{a}}$ and $S_{\bar{\bar{a}}}$ be two sets of blocked arcs, defined as follows:

$$S_{\bar{a}} = \{\bar{a}\} \cup (A \backslash \mathcal{A}_S(\boldsymbol{y})), S_{\bar{\bar{a}}} = \{\bar{\bar{a}}\} \cup (A \backslash \mathcal{A}_S(\boldsymbol{y}))$$

As $\boldsymbol{y}$ is minimal, these two sets of arcs correspond to feasible UMCFBP solutions. We use $\mathbf{x}^{S_{\bar{a}}}$ and $\mathbf{x}^{S_{\bar{\bar{a}}}}$ to denote the incidence vector associated with $S_{\bar{a}}$ and $S_{\bar{\bar{a}}}$, respectively. In other words, $x_a^{S_{\bar{a}}} = 1$ if and only if arc a is contained in the set $S_{\bar{a}}$. Similarly, $x_a^{S_{\bar{\bar{a}}}} = 1$ if and only if arc a is contained in the set $S_{\bar{\bar{a}}}$. We can easily observe that both vectors satisfy (5.14b) with equality, i.e., $\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a^{S_{\bar{\bar{a}}}} = x_a^{S_{\bar{a}}} = 1$. This implies that $b\mathbf{x}^{s_{\bar{a}}} = b\mathbf{x}^{s_{\bar{\bar{a}}}}$ and hence $b_{\bar{a}} = b_{\bar{\bar{a}}}$. By symmetry, we can deduce that there exists $\rho \in \mathbb{Z}$ such that $b_{\bar{a}} = b_{\bar{\bar{a}}} = \rho$ for every arc $\bar{a} \in \mathcal{A}_S(\boldsymbol{y}), \bar{\bar{a}} \in \mathcal{A}_S(\boldsymbol{y})$.
Let $a' \in A \setminus \mathcal{A}_S(\boldsymbol{y})$. We can state that there exists an arc $\bar{a} \in \mathcal{A}_S(\boldsymbol{y})$ such that $\mathcal{A}_S(\boldsymbol{y}) \setminus \{\bar{a}\} \cup \{a'\}$ does not contain a UMCF with a profit greater than Φ. Indeed, if such an event were to occur, it would imply the existence of an arc (s_k, t_k) for a commodity $k \in K$, which contradicts Hypothesis 1. To illustrate this statement, we consider the simplified graph depicted in Figure 5.5, featuring a single commodity denoted as $k_0 = (s_0, t_0, b_0, \Gamma_0)$, connected by two arcs, (s_0, u) and (u, t_0). We assume that these two arcs form a UMCF with a profit greater than Φ, thus implying that one of these arcs is the arc $\bar{a}$. Since no arc can directly link s_0 to t_0, for every vertex $v \in V \setminus \{s_0, u, t_0\}$, arc a' could potentially replace any arc linking v to s_0, u, or t_0. However, substituting any of the arcs in the UMCF with a' will not allow a UMCF of profit greater than Φ. In Figure 5.5, the arc a' is represented by one of the dashed lines. For sake of briefty, we did not represent the capacities, routing costs and blocker costs on the arcs.

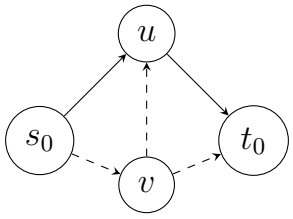


Figure 5.5: Graph with a single commodity $(s_0, t_0, b_0, \Gamma_0)$

We denote by $S_{a',\bar{a}}$ the UMCFBP solution that consists in removing arcs a' and $\bar{a}$ and we use $\mathbf{x}^{S_{a',\bar{a}}}$ to denote its associated incidence vector, i.e., $x_a^{S_{a',\bar{a}}} = 1$ if and only if arc a is contained in the set $S_{a',\bar{a}}$. This solution satisfies (5.14b) with equality, i.e., $\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a^{S_{a',\bar{a}}} = 1$. Therefore, we have $b\mathbf{x}^{S_{a',\bar{a}}} = b\mathbf{x}^{S_{\bar{a}}}$. This implies that $b_{a'} = 0$ and thus, $b = \rho g$. Which ends the proof. □

Proposition 25. *For every $\boldsymbol{y} \in ext(\mathcal{P}_{mcf})$, inequality $\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1$ defines a facet of $P(G, K)$ if and only if the MCF induced by vector $\boldsymbol{y}$ is minimal, i.e, for each arc $\bar{a} \in \mathcal{A}_S(\boldsymbol{y})$, decreasing its capacity results in an MCF with profit less than Φ.*

Proof. ($\Rightarrow$) Suppose there exists an arc $\bar{a} \in \mathcal{A}_S(\boldsymbol{y})$ for which decreasing its capacity results in an MCF with profit greater than Φ. Assuming that the capacity of the arc is 1, this process is equivalent to removing the arc from the graph. Accordingly, in this case, the following inequalities are valid: $\sum_{a \in \mathcal{A}_S(\boldsymbol{y}) \setminus \{\bar{a}\}} x_a \geq 1$ and $x_{\bar{a}} \geq 0$. Therefore, the

inequality $\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1$ cannot define a facet of $P(G, K)$ since it can be obtained as a linear combination of these inequalities.

($\Leftarrow$) We can apply the same reasoning as the one presented for $P_U(G, K)$. However, it is worth noticing that, in this case, the definition of minimality imposed by $\boldsymbol{y}$ is crucial. Indeed, let $a' \in A \setminus \mathcal{A}_S(\boldsymbol{y})$. If the MCF induced by vector $\boldsymbol{y}$ is not minimal in the sense that for each arc $\bar{a} \in \mathcal{A}_S(\boldsymbol{y})$, decreasing its capacity results in an MCF with profit less than Φ, then we cannot state that there exists an arc $\bar{a} \in \mathcal{A}_S(\boldsymbol{y})$ such that $\mathcal{A}_S(\boldsymbol{y}) \setminus \{\bar{a}\} \cup \{a'\}$ does not contain an MCF with a profit greater than Φ. To illustrate this, we consider the simple graph $G = (V, A)$ depicted in Figure 5.6, featuring a single commodity $k_0 = (s_0, t_0, b_0, \Gamma_0)$, with $b_0 = 10, \Gamma_0 = 1$. On each arc $a \in A$, we report the capacity c_a. The flow cost p_a is set to 0 for every arc $a \in A$. The target profit value Φ is set to 9. The blue lines represent an MCF with a profit equal to 10, which is accordingly greater than Φ. We remark that by removing the arc (v, t_0), the flow can be rerouted on the arc $a' = (v, u)$, leading to an MCF of profit equal to 10, which is greater than Φ. More precisely, this would imply that decreasing capacity of the arc (u, t_0) could lead to an MCF composed by the four arcs $\{(s_0, u), (u, t_0), (s_0, v), (v, t_0)\}$ with a profit greater than Φ. Therefore, it is necessary to impose a condition that $\mathbf{y}$ is minimal in the sense that for each arc $\bar{a} \in \mathcal{A}_S(\mathbf{y})$, decreasing its capacity results in an MCFP with profit less than Φ. Using this condition for the splittable variant, we can use the same proof by maximality as the one developed for the unsplittable variant (see Proposition 24) to prove that inequality $\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1$ defines a facet of $P(G, K)$.

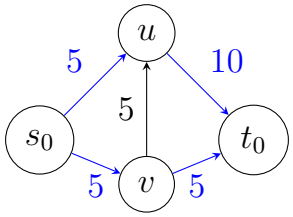


Figure 5.6: Graph with a single commodity $(s_0, t_0, b_0, \Gamma_0)$

□

5.5 Concluding remarks

In this chapter, we delve into the multi-commodity flow blocker problem, exploring both its splittable (MCFBP) and unsplittable (UMCFBP) variants. We conduct a comprehensive analysis of the complexity associated with both versions of the problem and establish a correlation between their solutions. Our approach begins with proposing a bilevel formulation to address these challenges, along with a single-level reformulation. We then present a more general integer linear programming (ILP) formulation with an exponential number of constraints. To enhance the model's robustness, we introduce a polyhedral study, aiming to describe the polyhedra of solutions for each problem and identify necessary and sufficient conditions for the inequalities to be facet-defining.

For future research, leveraging the bilevel formulation and its reformulation could prove beneficial in enhancing the proposed formulations by incorporating new valid inequalities.

The next chapter will be dedicated to the Branch-and-Cut algorithms used to solve the proposed formulations for the MCFBP and the UMCFBP.

Chapter 6

A Branch-and-Cut algorithm for the multicommodity flow blocker problem

Contents

In the preceding chapter, we presented a natural formulation tailored to address the multi-commodity flow blocker problem (MCFBP) in both its splittable and unsplittable variants. This formulation is characterized by an exponential family of constraints. The current chapter focuses on developing a specialized Branch-and-Cut algorithm for each variant of the problem. For the two families of constraints presented, namely the target profit inequalities(5.14b) *and the u-target profit inequalities*(5.16b)*, we delve into the associated separation problem, exploring its complexity and various separation strategies. Following this, we present experimental results to assess the effectiveness of our proposed approaches. Furthermore, we conduct a comparison analysis of our natural formulation against the current state-of-the-art technique for solving the MCFBP.*

6.1 The splittable multicommodity flow blocker problem

Let us consider the notations defined previously. Precisely, let $G = (V, A)$ be a directed graph with $m = |A|$ arcs and $n = |V|$ vertices. Each arc $a \in A$ is given a capacity $c_a \in \mathbb{R}_+$, a routing cost (or flow cost) $p_a \in \mathbb{R}_+$, representing the cost associated with routing a unit of flow and a blocker cost $r_a \in \mathbb{R}_+$, representing the cost for removing (or blocking) arc a. This graph is associated with a set of commodities K, where each commodity $k = (s_k, t_k, b_k, \Gamma_k)$ is characterized by a source $s_k \in V$, a destination $t_k \in V$, a bandwidth $b_k \in \mathbb{R}_+$, and a reward $\Gamma_k \in \mathbb{R}_+$. Finally, as previously, let Φ denote the target profit value.

We consider again the blocker vector $\mathbf{x}$ and the notations $\mathcal{A}(\mathbf{x})$ for the set of blocked arcs and $\mathcal{G}_{NB}(\mathbf{x})$ for the *non-blocked graph*.

Let $\boldsymbol{y} \in \mathcal{P}_{mcf}$ be a vector associated with an MCF of profit value greater than Φ in graph G, and let $\mathcal{A}_S(\boldsymbol{y}) \subseteq A$ be the subset of arcs routing the MCF. We recall that a natural formulation for the MCFBP reads as follows:

$$\min \quad \sum_{a \in A} r_a x_a \tag{6.1a}$$

$$\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1 \qquad \boldsymbol{y} \in ext(\mathcal{P}_{mcf}) \tag{6.1b}$$

$$x_a \in \{0, 1\} \qquad a \in A, \tag{6.1c}$$

The target profit inequalities (6.1b) are in exponential number. Accordingly, in order to solve the natural formulation (6.1), one needs to implement a Branch-and-Cut (B&C) algorithm where target profit inequalities are separated in the nodes of the branching tree for integer and fractional solutions. This exact algorithm requires defining a *relaxed master problem* (RMP) where the binary variables are replaced with continuous variables taking values between 0 and 1. In the initialization phase, only a subset of constraints are included in the RMP. To check that RMP solutions respect all the target profit inequalities or to determine one or more violated constraints which are then added to the RMP, we propose several separation procedures that will be described in the next sections.

6.1.1 Separation of target profit inequalities

Given a (fractional) solution $\mathbf{x} \in [0, 1]^m$ of the RMP in a B&C node, the separation problem for the target profit inequalities (6.1b) requires finding a multi-commodity flow defined by a vector $\boldsymbol{y}^* \in ext(\mathcal{P}_{mcf})$ such that:

$$\sum_{a \in A(\boldsymbol{y}^*)} x_a < 1, \tag{6.2}$$

or to prove that such a vector does not exist, i.e., that all target profit inequalities (6.1b) are satisfied by the solution $\mathbf{x}$. Thus, it is necessary to find a vector $\boldsymbol{y}^* \in ext(\mathcal{P}_{mcf})$ leading to the minimum value of the left-hand side of (6.2).

We distinguish two cases. The first one is for fractional RMP solutions and the second one is for integer RMP solutions.

For fractional RMP solutions $\boldsymbol{x}$, let $\boldsymbol{z} \in \{0,1\}^m$ be a vector of binary variables where each variable z_a is equal to 1 if and only if the arc a is in $A(\boldsymbol{y}^*)$, i.e., if the arc a is used to route a positive amount of flow in the multicommodity flow defined by $\boldsymbol{y}^*$. We recall that the total profit of this MCF is strictly greater than Φ, according to the definition of $\mathcal{P}_{mcf}$. The separation problem for the target profit inequalities can be modeled by the following ILP problem:

$$\min_{\mathbf{y}\in[0,1]^{d\times m},\boldsymbol{\lambda}\geq 0,\mathbf{z}\in\{0,1\}^m} \sum_{a\in A} z_a \cdot x_a \tag{6.3a}$$

$$\sum_{k\in K} y_{k,a}\, b_k \leq c_a, \qquad \forall\, a \in A \tag{6.3b}$$

$$\sum_{a\,\in\,\delta^+(u)} y_{k,a}\, b_k - \sum_{a\,\in\,\delta^-(u)} y_{k,a}\, b_k = \begin{cases} \lambda_k \text{ if } u = s_k, \\ 0 \text{ if } u \in V\backslash\{s_k,t_k\}, \\ -\lambda_k \text{ if } u = t_k \end{cases} \quad \forall k \in K, \forall u \in V \tag{6.3c}$$

$$y_{k,a} \leq z_a, \qquad a \in A,\ k \in K \tag{6.3d}$$

$$\sum_{k\in K} \Gamma_k\, \lambda_k - \sum_{a\in A} p_a\, y_{k,a}\, b_k \geq \Phi + 1. \tag{6.3e}$$

The objective function minimizes the left-hand size of (6.2). Constraints (6.3b) and (6.3c) are constraints of the MCFP, representing respectively the capacity constraints and the flow conservation constraints. The additional constraints "$y_{k,a} \leq z_a$" imposes to select an arc $a \in A$ if it routes a flow for a commodity $k \in K$ and constraint (6.3e) defines the minimum profit of the multicommodity flow. If the optimal solution value of this problem is strictly smaller than 1, then a target profit inequality maximally violated by $\mathbf{x}$ is found. Otherwise, no target profit inequalities are violated by $\mathbf{x}$.

To characterize the complexity of the separation problem associated with the target profit inequalities for fractional RMP solutions, we introduce the *minimum edge-cost flow problem* (MECFP). The MECFP is $\mathcal{NP}$-complete in its decision version, see Garey and Johnson [1979], and accordingly it is $\mathcal{NP}$-hard in its optimization version. Given an *arc-price* vector $\boldsymbol{\omega} \in \mathbb{Z}_+^m$ and a *flow-value bound* $B \in \mathbb{Z}_+$, the MECFP requires finding a minimum-price flow from a source $s \in V$ to a destination $t \in V$ with a flow value larger than or equal to B. It is worth noticing that the MECFP remains $\mathcal{NP}$-hard for $\boldsymbol{\omega} \in [0,1]^m$.

By reducing the MECFP to the separation problem, the next proposition characterizes the computational complexity of the latter one.

Proposition 26. *The separation problem for the target profit inequalities* (6.1b) *is $\mathcal{NP}$-hard for fractional solutions $\mathbf{x} \in [0,1]^m$ of the RMP.*

Proof. Starting from a MECFP instance, we set x_a equal to ω_a (a value in $[0,1]$) and the routing cost p_a equal to 0 for every arc $a \in A$. We focus on a single commodity $k_0 = (s,t,b_0,\Gamma_0)$, where $\Gamma_0 = 1$ and b_0 is a positive integer greater than or equal to B. The target profit value Φ is set to $B-1$. Since there is only one commodity, the

separation problem of inequalities (6.1b) results in finding a flow from s to t. Moreover, as $b_0 \geq B$, $\Gamma_0 = 1$ and $p_a = 0$ for all arcs $a \in A$, the total routing profit of this flow is greater than or equal to B, therefore guaranteeing that the value of the flow is also greater than or equal to B. Accordingly, once, the separation problem (6.3) is solved, its optimal solution $(\mathbf{y}, \mathbf{z})$ corresponds to an optimal MECFP solution. Indeed, since $\mathbf{y}$ defines a flow of value greater than or equal to B, all the MECFP constraints are satisfied, i.e., the capacity constraints, the flow conservation constraints and the requirement of having a flow value larger than or equal to B. Moreover, the variable values $\boldsymbol{z}$ correspond to the arcs with a minimum-price flow from s to t.

□

For an integer solution $\mathbf{x} \in \{0,1\}^m$, the separation problem can be restated as an MCFP and the next proposition characterizes its computational complexity:

Proposition 27. *The separation problem for the target profit inequalities* (6.1b) *can be performed in polynomial time for integer solutions* $\mathbf{x} \in \{0,1\}^m$ *of the RMP.*

Proof. Since $x_a \in \{0,1\}, \forall a \in A$, a violated target profit inequality can be found if and only if a multi-commodity flow defined by a vector $\boldsymbol{y} \in ext(\mathcal{P}_{mcf})$ exists such that the subset $A(\boldsymbol{y}) \subseteq A$ does not contain any blocked arcs. Accordingly, finding a violated target profit inequality leads to solving an MCFP in the non-blocked graph $\mathcal{G}_{NB}(\mathbf{x}) = \big(V, A \setminus \mathcal{A}(\mathbf{x})\big)$. In case the maximum profit of the MCF in the non-blocked graph is strictly greater than Φ, a multi-commodity flow defined by a vector $\boldsymbol{y} \in ext(P_{mcf})$ and associated to a violated target profit inequality is found. Otherwise, no target profit inequalities are violated by $\mathbf{x}$. Since the MCFP can be solved in polynomial time (see Tardos [1986]), the separation problem of the target profit inequalities (6.1b) can be solved in polynomial time for any integer RMP solution $\mathbf{x}$. □

6.1.2 Branch-and-Cut algorithm

In this section, we present a B&C algorithm to solve the MCFBP (Bentoumi et al. [2021]).

Indeed, in order to solve the natural formulation (6.1) with target profit inequalities (6.1b) for the MCFBP, one needs to implement a Branch-and-Cut algorithm with an efficient separation strategy for inequalities (6.1b). To determine a target profit inequality violated by an RMP solution $\mathbf{x}$, we propose the separation procedures described below.

Integer separation of target profit inequalities (6.1b)

One way to implement the B&C algorithm is to separate the target profit inequalities only for integer LP relaxation points, since modern MIP solvers guarantee that fractional points are cut off by the standard branching procedure strengthened with cutting plane mechanisms. As stated in Proposition 27, this can be done in polynomial time by solving an MCFP. To this end, one can solve the arc-formulation for the MCFP given by Model 5.3 in Chapter 5 using an LP Solver. However, in practice, due to the substantial

storage requirements associated with the arc-formulation, an alternative technique may be employed to solve the MCFP. This technique relies on a path formulation that is further solved using a column generation algorithm, described in the next paragraph.

A column generation algorithm to solve the MCFP Another formulation, referred to as the *path formulation*, can be designed for the MCFP, as well as for the UMCFP. As for the previous formulations, known as the arc formulations, we are given a directed graph $G = (V, A)$ with a set of commodities K. We introduce a new parameter P^k for every commodity $k \in K$. P^k represents the set of all distinct paths from s_k to t_k in the network. In the path formulation, we introduce a decision variable $y_{k,p}$ for each commodity $k \in K$ and each path $p \in P^k$. For every commodity $k \in K$, the variable $y_{k,p}$ represents the proportion of the bandwidth routed through path p. In the context of a splittable multicommodity flow problem, $y_{k,p}$ is a continuous variable ranging between 0 and 1, i.e., $y_{k,p} \in [0, 1]$. This implies that the bandwidth b_k can be distributed among multiple paths. Conversely, in the case of an unsplittable multicommodity flow problem, $y_{k,p}$ is binary, i.e., $y_{k,p} \in \{0, 1\}$. As mentioned in the previous chapter, in the UMCFP, a satisfied commodity has its entire bandwidth routed through a single path.

Therefore, the path formulation for the MCFP is given below.

$$\zeta(\text{MCFP}) \;=\; \max \quad \sum_{k \in K} \sum_{p \in P^k} (b_k \, \Gamma_k \, y_{k,p} - \sum_{a \in A(p)} b_k \, p_a \, y_{k,p}) \tag{6.4a}$$

$$\sum_{k \in K} \sum_{p \in P^k | a \in p} y_{k,p} \, b_k \leq c_a \qquad a \in A, \tag{6.4b}$$

$$\sum_{p \in P^k} y_{k,p} \leq 1 \qquad k \in K \tag{6.4c}$$

$$y_{k,p} \in [0, 1] \quad k \in K, p \in P^k. \tag{6.4d}$$

Constraints (6.4b) are the capacity constraints imposing that, for every commodity the total flow routed through an arc does not exceed the capacity of the arc. Constraints (6.4c) impose that, for all commodities $k \in K$, the total flow routed through all the paths P^k does not exceed the bandwidth b_k. Finally, constraints (6.4d) express the definition set of the path-flow variables $\mathbf{y}$.

Let us describe the column generation method to solve the path formulation (6.4) of the MCFP. We denote by (MCFP) the master LP (6.4) and its dual by (D-MCF). For every commodity $k \in K$, let $\mathcal{P}'^k \subseteq \mathcal{P}^k$ be a subset of the paths. We consider the restricted master LP (MCFP') with regard to $\mathcal{P}'^k$.

$$\zeta(\text{MCFP}') \;=\; \max \quad \sum_{k\in K}\sum_{p\in P'^k}(y_{k,p}\; b_k\; \Gamma_k - \sum_{a\in A(p)} y_{k,p}\; b_k\; p_a) \tag{6.5a}$$

$$\sum_{k\in K}\sum_{p\in P'^k|a\in p} y_{k,p}\; b_k \le c_a \qquad a\in A, \tag{6.5b}$$

$$\sum_{p\in P^k} yk,p \le 1 \qquad k\in K \tag{6.5c}$$

$$y_{k,p}\in[0,1] \quad k\in K, p\in P'^k. \tag{6.5d}$$

It is clear that a feasible solution for (MCFP') corresponds to a feasible solution for (MCFP). However, determining whether the solution of (MCFP') is optimal for (MCFP) requires further investigation. More precisely, if this is not the case, then additional variables are introduced to (MCFP') to enhance the solution value. This process can be done by using the dual restricted master LP (D-MCF'), presented below:

$$\zeta(\text{D}-\text{MCFP}') \;=\; \min \quad \sum_{a\in A} c_a\mu_a + \sum_{k\in K}\mu_k \tag{6.6a}$$

$$\mu_k + \sum_{a\in A(p)} b_k\mu_a \ge b_k(\Gamma_k - \sum_{a\in A(p)} p_a) \quad k\in K, p\in P'^k \tag{6.6b}$$

$$\mu_k\in[0,1] \qquad k\in K, \tag{6.6c}$$

$$\mu_a\in[0,1] \qquad a\in A. \tag{6.6d}$$

where μ_a are the dual variables associated to Constraints (6.4b), for every arc $a\in A$ and μ_k are the dual variables associated to Constraints (6.4c), for every commodity $k\in K$.

Let $\boldsymbol{\mu}$ be a solution of (D-MCF') that respects the dual constraints (6.6b) for all $p\in P^k, k\in K$, then $\boldsymbol{\mu}$ is a feasible and optimal solution for (D-MCF) (see Ahuja et al. [1993]).

Deciding whether Constraints (6.6b) are satisfied for all paths is the so-called *pricing problem* and can be solved efficiently by computing the shortest path for each commodity $k\in K$ between the source s_k and the destination t_k with a weight μ_a on every arc $a\in A$. If the shortest paths are all at least of weight 1, then Constraints (6.6b) are satisfied for all paths, otherwise, we have found a path that violated Constraints (6.6b).

An efficient separation strategy for fractional points can also help enhance the performance of a Branch-and-cut-based approach. We will now introduce some.

Exact separation of target profit inequalities (6.1b)

A second approach involves separating fractional solutions in addition to integer solutions. This can be done by solving the exact separation problem (see Model (6.3)).

Let $(\mathbf{y}^*, \mathbf{z}^*)$ be an optimal solution of Model (6.3). In case the optimal solution value is strictly smaller than 1, then we have detected a violated target profit inequality (6.1b) and the following cut is added to the RMP:

$$\sum_{a \in A(\boldsymbol{y}^*)} x_a \geq 1,$$

where $A(\boldsymbol{y}^*)$ is the set of arcs $a \in A$ for which $y^*_{k,a} > 0, \forall k \in K$.

However, as stated in Proposition 26, this problem is $\mathcal{NP}$-hard. Accordingly, the exact separation of target profit inequalities may be time-consuming.

Heuristic separation of target profit inequalities (6.1b)

We now propose an alternative method to separate fractional solutions, wherein a feasible solution of the separation problem (6.3) is considered instead of the optimal solution. To this end, we investigate a heuristic for the separation problem. This heuristic, referred to as `SP_Heu` is a simple and well-understood algorithm based on successive computations of shortest paths problems. We refer the interested reader to Chapter 1, where a detailed description of the shortest path problem (SPP) has been provided.

We recall that the separation problem of the target profit inequalities (6.1b) aims at finding a multicommodity flow defined by a vector $\boldsymbol{y} \in ext(\mathcal{P}_{mcf})$ such that inequality (6.2) is satisfied. The value $\sum_{a \in A(\boldsymbol{y})} x^*_a$ is referred to as the total use-arc-cost value.

Shortest-path based heuristic (`SP_Heu`) The general idea of the heuristic consists in constructing a multicommodity flow with a profit greater than Φ, by iteratively satisfying a set of commodities. We recall that a commodity $k \in K$ is satisfied if at least one unit of flow is routed from the source s_k to the destination t_k. Additionally, the rewards for commodities and routing costs on arcs are set up so that increasing the quantity of flow routed also increases the routing profit. Accordingly, the objective for each commodity is to route a quantity of flow that reaches the upper limit of b_k.

For each commodity $k \in K$, the algorithm identifies a set of paths to route the flow from s_k to t_k, while minimizing the total use-arc-cost. This process involves solving a set of shortest path problems (SPP) for each pair of source and destination nodes within the graph G. In this context, every arc in the graph is assigned a length denoted by l_a and a capacity denoted by c^*_a. This specific instance of the shortest path problem can be written as $\text{SPP}(G, s_k, t_k, \mathbf{c}^*, \mathbf{l})$, where $\mathbf{c}^*$ and $\mathbf{l}$ are the arc-capacity vector and arc-length vector, respectively.

Precisely, the algorithm starts with a profit $\Omega(G)$ equal to 0, i.e., all commodities are unsatisfied and the total use-arc-cost, denoted by $SumX$ is equal to 0. These values are incrementally updated during the construction of the multicommodity flow. At

the initialization phase, the arc capacity vector $\mathbf{c}^*$ and the arc-length vector $\mathbf{l}$ take respectively values of the initial arc-capacity vector $\mathbf{c}$ and vector $\mathbf{x}^*$. For each commodity $k \in K$, we consider a flow value F_k that is initially equal to 0 and that cannot exceed the bandwidth b_k. Therefore, while F_k does not exceed b_k, a path p^* between s_k and t_k is computed by solving SPP$(G, s_k, t_k, \mathbf{c}^*, \mathbf{l})$. When p^* is found, the arc-capacity vector $\mathbf{c}^*$ and the use-arc-cost vector are updated. Precisely, for every arc $a \in A(p^*)$, l_a is set to 0 and c^*_a is reduced by the value of the flow routed through the path, which is determined by the smallest value between two quantities: the minimum capacity among all arcs in p^* and the residual bandwidth, computed as $b_k - F_k$. On the other hand, every time p^* is found, the routing profit $\Omega(G)$ is increased by the routing profit of the path computed as $\min\{\min\{c_a : a \in A(p^*)\}, b_k - F_k\} \times (\Gamma_k - \sum_{a \in A(p^*)} p_a)$, where $\min\{\min\{c_a : a \in A(p^*)\}, b_k - F_k\}$ is the flow routed through the path. The total use-arc-cost is also increased by the total length of the path, i.e., $\sum_{a \in A(p^*)} l_a$.

The algorithm terminates under two conditions. First, it stops when a multicommodity flow with a profit greater than Φ and a total use-arc-cost less than 1 is found. In this scenario, the corresponding new inequality is added to the RMP. Alternatively, the algorithm also stops when the total use-arc-cost is greater than 1. In this case, no new inequality is added to the RMP. It is worth noticing that if all commodities are treated without finding an MCF with a total routing profit greater than Φ and such that $SumX > 0$, then as previously, no new inequality is added to the RMP.

In what follows, we give a detailed description of the heuristic, see Algorithm 4.

It is worth noticing that various strategies exist for defining an order to process the commodities. A notable approach involves sorting the commodities based on their rewards, prioritizing those with the highest rewards. Alternatively, commodities could be sorted according to their bandwidths. Another strategy might involve a balanced consideration, where a trade-off between the reward and bandwidth of each commodity is taken into account. Each of these sorting methods offers a distinct approach to optimizing the flow-routing process but in our experiments, we mainly focus on the first technique that prioritizes commodities by their rewards, in order to increase the routing profit.

6.2 The unsplittable multicommodity flow blocker problem

In alignment with Section 6.1, this section introduces a Branch-and-Cut algorithm tailored for addressing the unsplittable variant of the MCFBP, namely the UMCFBP.

Let $\boldsymbol{y} \in \mathcal{P}_{umcf}$ be a vector associated with a UMCF of profit value greater than Φ in graph G. Using the same notations as before, we recall that the natural formulation for the UMCFBP reads as follows:

Algorithm 4 Pseudo-code for the Shortest-Path Based Heuristic for the MCFBP `SP_Heu`$(G, K, \mathbf{x}^*)$

Input Data:
$G = (V, A)$: Directed graph
K: Set of commodities
$\mathbf{c}$: Initial arc-capacity vector
$\mathbf{x}^*$: fractional RMP solution
s_k, t_k, b_k, Γ_k: Source, destination, bandwidth and reward for each commodity $k \in K$
p_a : Routing cost on arc $a \in A$
procedure INITIALIZE
    $\Omega(G) = 0$
    $SumX = 0$
    $\mathbf{c}^* = \mathbf{c}$
    $\mathbf{l} = \mathbf{x}^*$
end procedure
for all $k \in K$ **do**
    $F_k = 0$
    while $F_k < b_k$ **do**
        $p^* = SPP(G, s_k, t_k, \mathbf{c}^*, \mathbf{l})$
        $F_k += \min\{\min\{c_a : a \in A(p^*)\}, b_k - F_k\}$
        for all $a \in A(p^*)$ **do**
            $l_a = 0$
            $c^*_a -= \min\{c_a : a \in A(p^*)\}$
            $SumX += l_a$
        end for
    end while
end for

$$\min \quad \sum_{a \in A} r_a x_a \tag{6.7a}$$

$$\sum_{a \in \mathcal{A}_S(\boldsymbol{y})} x_a \geq 1 \qquad \boldsymbol{y} \in ext(\mathcal{P}_{umcf}) \tag{6.7b}$$

$$x_a \in \{0,1\} \qquad a \in A, \tag{6.7c}$$

Inequalities (6.7b), referred to as the u-target profit inequalities, are in exponential number. Accordingly, in order to solve the natural formulation (6.7), one needs to implement a Branch-and-Cut (B&C) algorithm where u-target profit inequalities are separated in the nodes of the branching tree for integer and fractional solutions. As previously, this exact algorithm requires defining a *relaxed master problem* (RMP) where the binary variables are replaced with continuous variables taking values between 0 and 1. In the initialization phase, only a subset of constraints are included in the RMP. To check that RMP solutions respect all the u-target profit inequalities or to determine one or more violated constraints which are then added to the RMP, we propose several separation procedures that will be described subsequently.

In the next section, we present the separation problem associated with the u-target profit inequalities.

6.2.1 Separation of u-target profit inequalities

Given a (fractional) solution $\mathbf{x} \in [0,1]^n$ of the RMP in a B&C node, the separation problem for the u-target profit inequalities (6.7b) requires finding an unsplittable multicommodity flow defined by a vector $\boldsymbol{y}^* \in ext(\mathcal{P}_{umcf})$ such that:

$$\sum_{a \in A(\boldsymbol{y}^*)} x_a < 1, \tag{6.8}$$

or to prove that such a vector does not exist, i.e., that all u-target profit inequalities (6.7b) are satisfied by the solution $\mathbf{x}$. Thus, it is necessary to find a vector $\boldsymbol{y}^* \in ext(\mathcal{P}_{umcf})$ leading to the minimum value of the left-hand side of (6.8).

Let $\boldsymbol{z} \in \{0,1\}^m$ be a vector of binary variables where each variable z_a is equal to 1 if and only if the arc a is in $A(\boldsymbol{y}^*)$, i.e., if the arc a is used to route a positive amount of flow in the unsplittable multicommodity flow defined by $\boldsymbol{y}^*$. We recall that the total profit of this UMCF is strictly greater than Φ, according to the definition of $\mathcal{P}_{umcf}$.

The separation problem for the u-target profit inequalities can be modeled by the following ILP problem:

$$\min_{\mathbf{y} \in \{0,1\}^{d \times m}, \boldsymbol{\lambda} \geq 0, \mathbf{z} \in \{0,1\}^m} \Big\{ \sum_{a \in A} z_a \cdot x_a : (6.3b), (6.3c), (6.3d), (6.3e). \Big\} \tag{6.9}$$

Model (6.9) is the same as Model (6.3), which represents the exact separation problem for target profit inequalities (6.1b). Indeed, the two models have the same objective function and constraints. However, they differ in the variable set $\mathbf{y}$, representing the flow.

The next proposition characterizes the computational complexity of the separation problem for the u-target profit inequalities (6.7b).

Proposition 28. *The separation problem for the u-target profit inequalities* (6.7b) *is $\mathcal{NP}$-hard for fractional and integer solutions* $\mathbf{x}$ *of the RMP.*

Proof. The decision problem of the separation problem for inequalities (6.7b) asks for finding an unsplittable multi-commodity flow with a total profit exceeding $\Phi + 1$. This problem is $\mathcal{NP}$-Complete as stated previously. Accordingly, the optimization version of the problem is $\mathcal{NP}$-hard. □

6.2.2 Branch-and-Cut algorithm

In this section, we present a Branch-and-Cut algorithm to solve the UMCFBP. Indeed, in order to solve the natural formulation (6.7) with u-target profit inequalities (6.7b), one needs to implement a Branch-and-Cut algorithm with an efficient separation strategy for Inequalities (6.7b). To determine a u-target profit inequality violated by an RMP solution $\mathbf{x}$, we propose the separation procedures described below.

Integer separation of u-target profit inequalities (6.7b)

One way to implement the B&C algorithm is to separate the u-target profit inequalities only for integer LP relaxation points $\mathbf{x} \in \{0,1\}^n$, since modern MIP solvers guarantee that fractional points are cut off by the standard branching procedure strengthen with cutting plane mechanisms. This can be done by solving a UMCFP. Indeed, since $x_a \in \{0,1\}, \forall a \in A$, a violated u-target profit inequality can be found if and only if an unsplittable multi-commodity flow defined by a vector $\boldsymbol{y} \in ext(\mathcal{P}_{umcf})$ exists such that the subset $A(\boldsymbol{y}) \subseteq A$ does not contain any blocked arcs. Accordingly, finding a violated u-target profit inequality leads to solve a UMCFP in the non-blocked graph $\mathcal{G}_{NB}(\mathbf{x}) = (V, A \setminus \mathcal{A}(\mathbf{x}))$. In case the maximum profit $\Psi(\mathcal{G}_{NB}(\mathbf{x}))$ is strictly greater than Φ, an unsplittable multicommodity flow defined by a vector $\boldsymbol{y} \in ext(P_{umcf})$ and associated to a violated u-target profit inequality is found. Otherwise, no u-target profit inequalities are violated by $\mathbf{x}$. However, as the UMCFP is $\mathcal{NP}$-Hard, the separation of integer LP relaxation points may considerably impact the overall performance of the B&C algorithm. Subsequently, we propose an algorithm designed to efficiently generate u-target profit inequalities, using a column-generation-based heuristic approach.

As for the MCFP, an alternative formulation can be designed for the UMCFP using a path-based approach. Specifically, the path formulation previously introduced for the MCFP remains applicable for its unsplittable variant, the UMCFP. However, there is a distinction in the variable set $\mathbf{y}$, which becomes a vector of binary variables, i.e, for every commodity $k \in K$ and for every path $p \in P^k$, we have $y_{k,p} \in \{0,1\}$.

Therefore, the path formulation for the UMCFP reads as follows:

$$\zeta(\text{UMCFP}) \;=\; \max \quad \sum_{k\in K}\sum_{p\in P^k}(y_{k,p}\; b_k\; \Gamma_k - \sum_{a\in A(p)} y_{k,p}\; b_k\; p_a) \tag{6.10a}$$

$$\sum_{k\in K}\sum_{p\in P^k|a\in p} y_{k,p}\; b_k \le c_a \qquad a\in A, \tag{6.10b}$$

$$\sum_{p\in P^k} y_{k,p} \le 1 \qquad k\in K, \tag{6.10c}$$

$$y_{k,p}\in\{0,1\} \quad k\in K, p\in P^k. \tag{6.10d}$$

Achieving optimality for Model (6.10) would require the implementation of a Branch-and-Price algorithm (refer to Schrijver [2003]). Conversely, our approach involves solving the linear relaxation of the model and subsequently applying a rounding algorithm, presented in the next paragraph, to derive integer solutions. This approach yields a feasible solution for the UMCFP.

Rounding algorithm The rounding procedure presented is known as *randomized rounding* (Krolikowski et al. [2021b]), wherein paths are chosen based on probabilities derived from the optimal solution of the linear relaxation of Model (6.10). The objective of this procedure is to select paths randomly, with probabilities proportional to their respective linear relaxation values. More precisely, for a set of commodities $K' \subseteq K$ processed in a random order, a path is randomly selected for every commodity $k \in K'$, where the probability of selecting a path is proportional to the linear relaxation value of $y_{k,p}, k \in K', p \in P^k$. It is worth noticing that each time a path is chosen to satisfy a commodity, the capacity of the arcs in this path is updated, i.e., reduced by the bandwidth. Furthermore, the total profit value of the current UMCF is computed. The algorithm is iterated for a specified number of times to compare multiple solutions, and the best solution value is maintained, representing the combination of paths resulting in a multicommodity flow with the highest profit value.

Hence, a method for deriving u-target profit inequalities for integer LP relaxation points $\mathbf{x} \in \{0,1\}^m$ involves heuristically solving a UMCFP in the non-blocked graph $\mathcal{G}_{NB}(\mathbf{x})$. This is achieved by first solving the LP relaxation of the path formulation (6.10) for the UMCFBP using a column generation algorithm and then applying a rounding procedure to obtain an integer solution. If an unsplittable multicommodity flow with a profit greater than Φ is identified, a new target-u profit inequality associated with this UMCF is added to the RMP. On the other hand, if such a UMCF is not found, the UMCFBP must be solved to optimality to ensure that no u-target profit inequalities are violated by $\mathbf{x}$.

An effective separation strategy for fractional points can also significantly enhance the performance of the algorithm. In what follows, we introduce two distinct strategies aimed at separating fractional points.

Exact separation of u-target profit inequalities (6.7b)

This section delves into the separation of fractional solutions by addressing the exact separation problem of u-target profit inequalities (6.7b).

Let $(\mathbf{y}^*, \mathbf{z}^*)$ be an optimal solution of Model (6.9). In case the optimal solution value is strictly smaller than 1, then we have detected a violated u-target profit inequality (6.7b) and the following cut is added to the RMP:

$$\sum_{a \in A(\boldsymbol{y}^*)} x_a \geq 1,$$

where $A(\boldsymbol{y}^*)$ is the set of arcs $a \in A$ for which $y^*_{k,a} > 0, \forall k \in K$.

However, solving Model (6.9) may be time-consuming. Therefore, in the next section we propose an algorithm to heuristically separate fractional solutions.

Heuristic separation of u-target profit inequalities (6.7b)

We now present two heuristics to solve the exact separation problem of the u-target profit inequalities, i.e., Model (6.9). The first heuristic, denoted as `U_SP_HEU`, is adapted from the heuristic `SP_HEU` presented in Section 6.1 for the MCFBP. The second heuristic, denoted by `U_CSP_HEU`, is an adapted variant of the *Lagrangian Relaxation based aggregated cost* (LARAC) algorithm for the constrained shortest path problem.

`U_SP_HEU` **heuristic** This heuristic employs the same logic as the `SP_HEU` heuristic designed for the MCFBP. However, for the `U_SP_HEU` heuristic, the flow for each commodity is routed through a single path. A detailed presentation of the algorithm is given by Algorithm 5.

Algorithm 5 Pseudo-code for the Shortest-Path Based Heuristic for the UMCFBP `U_SP_Heu`$(G, K, \mathbf{x}^*)$

Input Data:
$G = (V, A)$: Directed graph
K: Set of commodities
$\mathbf{c}$: Initial arc-capacity vector
$\mathbf{x}^*$: fractional RMP solution
s_k, t_k, b_k, Γ_k: Source, destination, bandwidth, and reward for each commodity $k \in K$
p_a : Routing cost on arc $a \in A$
procedure INITIALIZE
    $\Omega(G) = 0$
    $SumX = 0$
    $\mathbf{c}^* = \mathbf{c}$
    $\mathbf{l} = \mathbf{x}^*$
end procedure
for all $k \in K$ **do**
    $\mathbf{lk} = \mathbf{l}$
    $Gk = (V, Ak)$ ▷ Ak is the set of arcs $a \in A$ with $c_a \geq b_k$
    $p^* = SPP(Gk, s_k, t_k, \mathbf{c}^*, \mathbf{lk})$
    for all $a \in A(p^*)$ **do**
        $l_a = 0$
        $c^*_a - = b_k$
        $SumX + = l_a$
    end for
end for

Constrained shortest-path based heuristic (`U_CSP_Heu`) A second heuristic strategy to address the separation problem of u-target profit inequalities focuses on determining an optimal path for each commodity. This entails finding the shortest path that satisfies a specific constraint, which corresponds to a problem known in the literature as the constrained shortest path problem.

We first present the *constrained shortest path problem* (CSPP) also known as the shortest weight-constrained path, see e.g. Garey and Johnson [1979] where it has been shown to be $\mathcal{NP}$-hard. Given a graph $G = (V, A)$ where a length l_a and a weight ω_a is affected to every arc $a \in A$, the CSPP aims at finding a path p between a source $s \in V$ and a destination $t \in V$ that minimizes the total length of the path, i.e., the sum of the lengths on the arcs constituting the path, and such that the total weight of the path, i.e, the sum of the weights on the arcs constituting the path, does not exceed a given value W. The CSPP is a well-known optimization problem and many algorithms have been designed to exactly or heuristically solve this problem. In Juttner et al. [2001], the authors present the LARAC algorithm, a heuristic to solve the CSP, referred to as the *delay constrained least cost path problem*. This heuristic relies on the *Lagrange Relaxation*, which is a common technique used for finding lower bounds in a minimization problem. This technique has been introduced in Held and Karp [1970] for the Traveling Salesman problem. The main idea of this technique is to remove a set of constraints

defined as hard constraints and add them into the objective function as penalization terms for violating the hard constraints.

In what follows, we describe `U_CSP_Heu`, a heuristic that is based on the LARAC algorithm for the CSP, to solve the separation problem of the u-target profit inequalities. This algorithm uses the LARAC algorithm at each iteration. To explain `U_CSP_Heu`, we recall that the separation problem of the u-target profit inequalities aims to find an unsplittable multi-commodity flow with a profit greater than the target profit value Φ and such that inequality (6.2) is satisfied. We first focus on a particular case of the UMCFBP where we consider only one commodity k_0. It is clear that if the bandwidth of this commodity can be routed through a path of the graph, with a profit greater than Φ and while ensuring inequality (6.2), then a violated u-target profit inequality is found and added to the RMP. We now study the general case of $|K| \geq 1$. We can treat all commodities iteratively. For each commodity $k \in K$, we find a path in the graph G considering only the arcs with a capacity greater than b_k. This path can be found by solving the LARAC algorithm. Then, as for the `U_SP_Heu` algorithm, every time a path for a commodity is found, we upgrade the value of the total routing profit and the arc-capacity vector. At initialization, we also consider a bound corresponding to $SumX$ (see Algorithm 5) set to value 1. At each iteration, when a path is found, this bound is decreased with the sum of the values x^*_a of the arcs a constituting the path, then vector $\mathbf{x}^*$ is updated, i.e., the values of x^*_a are then set to 0. The algorithm stops under several conditions. First, it stops when $SumX$ reaches a zero or a negative value. In this case, no cuts are found. A second condition is related to the total routing profit. If it exceeds the target profit value Φ, then a new cut is found and added to the RMP. Finally, if all commodities are treated without finding a UMCF with a total routing profit greater than Φ and such that $SumX > 0$, then as previously, no cut is added to the RMP.

6.3 Computational results

In this section, we present the results of our computational campaign. The aim is to assess the performance of the ILP natural formulations for the MCFBP and the UMCFBP, presented in this thesis, i.e., Model (6.1) and Model (6.7). These models feature an exponential number of constraints. Accordingly, they are solved via Branch-and-Cut algorithms.

For each formulation, our computational campaign has two goals. The first one is to assess the best configuration of the Branch-and-Cut algorithms for the natural formulations. The second is to determine, for each formulation, the maximum size of the instances that can be solved to proven optimality using the exact method proposed in this paper to solve the MCFBP and the UMCFBP. Specifically, we investigate the practical computational difficulty of the main features of the instances. We also compare the performance of the ILP formulations proposed against the direct use of a bilevel solver, which corresponds to the current state-of-the-art approach. This technique involves employing an online-accessible solver designed for bilevel programs. This solver is the same as the one used to solve the maximum flow blocker problem in Chapter 4.

The experiments are conducted on a processor Intel Core i5-3340M CPU of 2.70GHz ×

4. To tackle the ILP formulations, we use `CPLEX` in `C++` and its `CONCERT TECHNOLOGY`. In our experiments, all computations are performed in a single-thread mode with default values for all `CPLEX` parameters.

6.3.1 Benchmark set of instances

To the best of our knowledge, this thesis presents the first exact methods to solve the MCFBP and the UMCFBP and no instances can be found in the literature. Hence, we have generated synthetic instances and derived others from datasets available in the literature to design networks.

The MCFBP and UMCFBP instances are characterized by three main components: *i*) the *graph-structural* features, *ii*) the *commodity-related* features and *iii*) the *flow-blocker* features.

Constructing a diverse and representative collection of instances for analysis poses a significant challenge. This section details the specific criteria and choices implemented in developing our benchmark set of instances for the MCFBP and the unsplittable variant, i.e., the UMCFBP. The structural characteristics of the instances are based on the input directed graph $G = (V, A)$. The primary graph-structural features are the number of vertices $n \in V$ and the number of arcs $m \in A$. The arc density of the graph, denoted as $d(G) = \frac{|A|}{n^2-n}$ is another crucial feature. Furthermore, in the context of MCFBP and UMCFBP, each arc is associated with specific capacities and routing costs. It is also crucial to consider the features related to commodities. In this context, each commodity within these instances is characterized by four fundamental elements: its source node, its destination node, the required bandwidth, and the reward value for transporting a unit of flow from the source to the destination. These elements, along with the arc-capacities and routing costs play a vital role in computing the maximum profit $\Omega(G)$ of an MCF in the graph.

The flow-blocker features of MCFBP and UMCFBP instances consist in determining the blocker cost $r_a \in \mathbb{Z}_+$ for each arc $a \in A$ and the target flow $\Phi \in \mathbb{Z}_+$. We have chosen to determine the target flow as a percentage $\lambda \in [0, 1]$ of the maximum profit value $\Psi(G)$ in the given instances. By opting for this approach, we can conduct an analysis of the computational complexity of the instances based on the maximum profit allowed in the non-blocked graph, which is determined as a percentage of the maximum profit in the original graph G. It is worth noticing that if $\lambda = 0$, the value of the target flow Φ is equal to 0, which implies that no MCF should remain in the non-blocked graph. On the other hand, if $\lambda = 1$, the value of the target profit Φ is equal to the maximum profit value in G, i.e., $\Phi = \Psi(G)$, implying that there is no need to remove any arcs from G. The distribution of the blocker costs on the arcs determines the value of the optimal solution for the MCFBP and the UMCFBP.

As far as the graph-structural features of MCFBP and UMCFBP instances are concerned, we consider three classes of instances: *i*) `SNDLIB` graphs, *ii*) `TELECOM` graphs, *iii*) `SYNTHETIC` graphs. The first class gathers instances obtained from the literature. The second class are instances provided by Huawei and the last class was generated from a random graph generator that we implemented in `Python 3.8.10`.

1. `SNDLIB` instances are sourced from SNDlib (Survivable Network Design Library), a well-known repository for test instances in network design, particularly in telecommunications. SNDlib provides a collection of network instances that include graphs and associated traffic demand scenarios. These instances are based on realistic network topologies and demand scenarios, making them valuable for practical network optimization studies. The library and detailed information about these instances can be accessed on the website[1]. For the purpose of our experiments, we selected specific characteristics from the SNDlib instances as follows: directed demands, directed links, explicit link capacities (as opposed to linear, modular, or single modular link capacities), a fixed-charge cost model and an admissible path model considering all paths without an explicit list. Furthermore, we included all survivability models in our analysis. Figure 6.1 provides an example of a typical network topology from SNDlib. Moreover, we considered two distinct sets of instances, each comprising 27 distinct networks. The first set was used for experiments involving continuous routing, while the second set was dedicated to discrete (unsplittable) routing. This selection and categorization were based on the inherent characteristics of the SNDlib instances. We refer the interested reader to Orlowski et al. [2007] for further details on these instances.

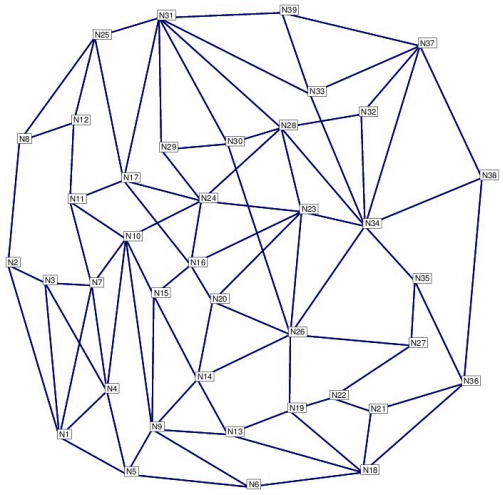


Figure 6.1: Example of a network topology from SNDlib

2. The `TELECOM` instances consist of real-world telecommunications network data along with traffic demands, provided by Huawei. This dataset is divided into two distinct categories: `TELECOM_A`, `TELECOM_B` and `TELECOM_C`. These instances represent graphs of type IPRAN (IP Radio Access Network), which is a special class of telecommunication networks having a specific structure. More precisely, in telecommunications, IPRAN graphs are pivotal and depict the network's structure connecting radio access equipment to the main IP network, which is crucial for efficient data transmission. Figure 6.2 presents a typical example of an IPRAN (IP Radio Access Network) network topology, illustrating the characteristic structure of this type of telecommunications network. This class comprises a total of 45 instances, with 15 instances per type. The instances of `TELECOM_A` consist

[1] `http://sndlib.zib.de/home.action`

of graphs with 71 vertices, averaging 500 arcs, and encompassing 60 commodities. For `TELECOM_B` instances, the graphs feature 1271 vertices, an average of 2500 arcs, and include 30 commodities. Lastly, the `TELECOM_C` instances are characterized by graphs with 5021 vertices, averaging 8000 arcs, and containing 10 commodities.

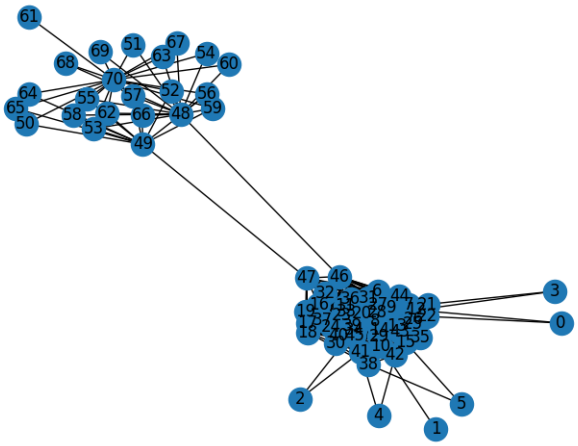


Figure 6.2: Example of an IPRAN network topology

3. The `SYNTHETIC` class consists of Erdős-Rényi random graphs denoted as $G(n, p)$, where the parameter p determines the probability of including each arc. In this case, the value of p is set to 0.5, meaning that each arc has a 50% chance of being included in the graph. We consider a large set of instance sizes based on the number of vertices n in the graph ranging from 20 to 300. Specifically, n takes the following values: 20, 40, 50, 70, 100, 130, 150, 200, 250 and 300. The size of the instances also varies depending on the arc densities $d(G)$, which range from 0.1 to 0.9 with a step size of 0.1. For each combination of n and $d(G)$, we create a total of 5 distinct Erdős-Rényi graphs using different random seed generators. This dataset is divided into two distinct categories, `SYNTHETIC_SAME` and `SYNTHETIC_DIFF`. These two classes of instances have the same networks, i.e., the same graph with a different set of commodities. In `SYNTHETIC_SAME`, all commodities have the same source and different destinations, while in `SYNTHETIC_DIFF`, all commodities have different sources and destinations. For each commodity $k \in K$, the source s_k and the destination t_k are randomly selected between the vertices of the graph such that s_k and t_k are linked by at least three consecutive arcs. The number of commodities within each of these classes can vary, encompassing options of 5, 10, 20, 25, 30, 40, 50 or 60 commodities. The arc-capacities are generated by drawing a number from the range [1, 49] (resp. [1, 3]) uniformly at random and rounding it to the nearest integer value. The bandwidth of the commodities are also randomly generated by selecting a number from the discrete uniform distribution on $[1, 49]$.

For each instance, we assign a routing cost to every arc and a blocker cost. These costs are determined by randomly selecting numbers from respective discrete uniform distributions. Specifically, for all instances, the routing cost and the blocker cost for each arc are generated from a distribution within the range of $[1, 49]$. Additionally, for every commodity, we associate a reward value, which is also randomly selected from a

discrete uniform distribution, defined within the range $[50, 100]$. This selection helps guarantee that routing a single unit of flow results in an increase in the profit.

In the remainder of this paper, all computing times are expressed in seconds. We impose a CPU time limit of 600 seconds for every instance. If this time limit is exceeded, it is reported as $t.l.$ in the computational results.

6.3.2 Computational performance of the Branch-and-Cut for the MCFBP

In this section, we study and discuss the computational performance of Model (6.1) for the MCFBP. The aim of this section is to determine the best Branch-and-Cut algorithm to solve the natural formulation for the MCFBP.

The natural formulation is solved using a Branch-and-Cut algorithm where target profit inequalities (6.1b) are separated at every node of the branching tree and added to the RMP. It is worth noticing that the RMP is initialized with one constraint that is associated with the maximum-profit multicommodity flow in G. This MCF has, by definition, a profit greater than Φ. We recall that we are interested in separating target profit inequalities both for integer and fractional solutions. As explained previously, the separation of target profit inequalities can be done in polynomial time by solving an MCFP, see Model (5.3) of Chapter 5, using `CPLEX` LP solver. For fractional solutions, the separation problem of target profit inequalities is $\mathcal{NP}$-hard (see Proposition 26). Consequently, an efficient separation procedure is crucial. In light of this, we have explored several strategies to separate target profit inequalities (6.1b). The first approach involves separating fractional solutions by solving the exact separation problem (see Model (6.3)). Additionally, alternative methods have been considered, wherein a feasible solution of the separation problem (6.3) is obtained instead of the optimal solution. To this end, we can use the shortest-path-based heuristic `SP_Heu` presented in Section 6.1.2.

Therefore, we propose three Branch-and-Cut algorithms to solve the natural formulation (6.1). The first one, denoted by `INT_MCFBP`, consists in separating only integer unfeasible points. Indeed, this is possible since modern MIP solvers guarantee that fractional points are cut off by the standard branching procedure strengthened with cutting plane mechanisms. However, an efficient separation strategy for fractional points can enhance the overall performance of the Branch-and-Cut algorithm. Accordingly, we investigate both types of separations in the two other algorithms. The second Branch-and-Cut algorithm, denoted by `EXACT_MCFBP`, consists in separating, in addition to the integer points, fractional points by solving the exact separation problem, i.e., Model (6.1b). The last algorithm, denoted by `HEU_MCFBP` consists in solving the heuristic `SP_Heu` instead of the exact separation problem.

Table 6.1, 6.2 and 6.3 compare the performance of the three Branch-and-Cut algorithms on `SNDLIB` instances for three target profit values defined by $\lambda = 0.2$, $\lambda = 0.6$ and $\lambda = 0.9$. Each row in the table corresponds to instances from `SNDLIB` class grouped by the size, defined by the number of vertices n in the graph, the number of arcs m, the number of commodities d, and the value λ to define the target profit. The column #

reports the total number of instances per row. For each configuration of the Branch-and-Cut algorithm, we report the following values: the number of instances solved to proven optimality (#opt), the average (avg.) and maximum (max) computing time (time) in seconds, the average number of constraints generated to separate integer infeasible points (#lazy), the average number of nodes in the branching tree (nodes), the average and maximum optimality gap value (opt gap). The optimality gap represents the percentage discrepancy between the current solution (LB) and the best possible solution (UB) and it is computed as $100 \times \frac{UB-LB}{UB}$. For `EXACT_MCFBP` and `HEU_MCFBP`, we also report the average number of constraints generated to separate fractional infeasible points (#user).

In Tables 6.2 and 6.3, we observe that the strategy of separating both integer and fractional infeasible points by solving the exact separation problem (6.3) yields better performance for the natural formulation on `SNDLIB` instances. Indeed, the `EXACT_MCFBP` configuration successfully solves 29 instances to proven optimality, surpassing `INT_MCFBP` and `HEU_MCFBP` by 2 instances. As expected, compared to `INT_MCFBP`, the separation of fractional infeasible points contributes to a reduction in the number of constraints required for separating integer infeasible points. For instance, for the smallest instances of 26 vertices with $\lambda = 0.6$, the number of integer infeasible points drops from 854 for `INT_MCFBP` to 163 for `EXACT_MCFBP`. This shows that the separation of fractional unfeasible points contributes to reaching the optimal solution faster. This is further evidenced by the considerably reduced number of nodes in the branching tree for `EXACT_MCFBP`, with only 77 nodes instead of 525 for `INT_MCFBP`.

However, with `HEU_MCFBP`, no improvement in total computing time is observed compared to `INT_MCFBP`. Furthermore, separating fractional infeasible points in this case does not contribute to reaching the optimal solution faster, as the number of nodes in the branching tree remains comparable to that of `INT_MCFBP`. In some instances, this number surpasses that of `INT_MCFBP`. For example, in graphs with 26 vertices, the branching tree comprises 687 nodes with `HEU_MCFBP` compared to 525 with `INT_MCFBP`. On the other hand, when fractional solutions are separated by solving the exact separation problem (6.3), `EXACT_MCFBP` successfully solves two additional instances to proven optimality: one with 26 vertices and 84 arcs, and another with 27 vertices and 102 arcs. Given the scale of these instances, the computing time required to solve the exact separation problem is reasonable (approximately 0.1 seconds). Meanwhile, the heuristic algorithm `SP_HEU` requires only a few milliseconds. However, the number of cuts needed to reach the optimal solution may increase and occasionally deviate from the optimal solution, leading to a decrease in the overall performance of the algorithm. This scenario is clearly demonstrated in instances with 26 vertices, where 11335 cuts are generated. For larger instances, the time saved by solving the heuristic might facilitate a quicker convergence to the optimal solution, as indicated by the value of the optimality gap (opt gap). For instance, in graphs with 39 vertices and 122 arcs and with $\lambda = 0.2$, the opt gap is 91% for `HEU_MCFBP`, compared to 96% for `EXACT_MCFBP`. This discrepancy becomes significant with $\lambda = 0.6$, where the average optimality gap is 74% for `HEU_MCFBP` and 81% for `EXACT_MCFBP`.

Moreover, we observe that for all configurations of the algorithm, namely `INT_MCFBP`, `EXACT_MCFBP`, and `HEU_MCFBP`, the computing time increases proportionally with the value of the target profit defined as a percentage λ of the maximum MCF profit value

in the graph. Consequently, a high value for λ suggests that a significant portion of the MCF profit is allowed in the graph, leading to a lower blocker cost. Additionally, it implies a smaller number of MCFs to block, or equivalently a smaller number of extreme points in $\mathcal{P}_{mcf}$, thus reducing the number of target profit inequalities (6.1b) in the natural formulation (6.1). On the contrary, decreasing λ results in a lower target profit value, necessitating the blocking of more MCFs. This is characterized by an increased number of extreme points in $\mathcal{P}_{mcf}$ and a greater number of target flow inequalities in Model (6.1), which contributes to the increased complexity of the model, and consequently, the computing time. For example, in a graph of 39 vertices and 122 arcs with $\lambda = 0.9$, the number of instances solved to proven optimality is 9 out of 9 (100% of the instances). However, with $\lambda = 0.6$, this number drops to 1, and with $\lambda = 0.2$, no instances are solved to proven optimality within the time limit.

					INT_MCFBP						
						time		# lazy	nodes	opt gap	
n	m	d	λ	#	# opt	avg.	max	avg.	avg.	avg.	max
26	84	15	0.2	9	0	t.l.	t.l.	907.78	316.89	90.95	98.97
			0.6	9	1	566.56	t.l.	854.89	525.56	58.78	84.24
			0.9	9	9	1.33	4.00	45.78	31.44	0.00	0.00
27	102	16	0.2	6	0	t.l.	t.l.	762.67	240.67	93.21	97.61
			0.6	6	0	t.l.	t.l.	756.33	349.67	84.45	90.64
			0.9	6	6	4.33	15.00	61.50	38.00	0.00	0.00
39	122	23	0.2	9	0	t..L.	t.l.	586.56	169.67	91.97	99.65
			0.6	9	1	536.33	t.l.	514.44	243.22	74.66	96.32
			0.9	9	8	69.00	t.l.	100.00	89.33	5.85	52.63
	172	23	0.2	3	0	t.l.	t.l.	489.67	73.33	98.77	99.74
			0.6	3	0	t.l.	t.l.	487.00	102.67	96.48	99.74
			0.9	3	2	231.33	t.l.	256.00	99.67	24.19	72.58
Total				81	27						

Table 6.1: Performance of INT_MCFBP Branch-and-Cut algorithm used to solve Model (6.1) on SNDLIB instances

In Table 6.1, Table 6.2 and Table 6.3, we can see that the natural formulation does not achieve proven optimality for all instances from SNDLIB, successfully solving 29 out of 81 instances, which corresponds to approximately 36%. This is mainly due to the intricate structure of the networks combined with a large number of demands. Subsequently, we will delve into the performance of the formulation on the TELECOM class of instances.

					EXACT_MCFBP							
					# opt	time		# lazy	# user	nodes	opt gap	
n	m	d	λ	#	# opt	avg.	max	avg.	avg.	avg.	avg.	max
26	84	15	0.2	9	0	t.l.	t.l.	224.89	3538.67	73.44	68.97	97.71
			0.6	9	2	580.00	t.l.	163.44	2347.00	77.22	53.24	79.22
			0.9	9	9	4.00	18.00	25.00	63.67	8.00	0.00	0.00
27	102	16	0.2	6	0	t.l.	t.l.	184.50	3287.67	53.17	83.22	99.40
			0.6	6	1	567.00	t.l.	134.33	1811.50	44.33	78.93	99.51
			0.9	6	6	6.33	22.00	29.50	110.50	14.83	0.00	0.00
39	122	23	0.2	9	0	t.l.	t.L.	162.78	1540.89	28.00	96.11	100.00
			0.6	9	1	546.89	t.L.	127.44	982.22	30.33	81.70	100.00
			0.9	9	8	69.44	t.L.	27.33	217.33	12.44	2.66	23.93
	172	23	0.2	3	0	t.l.	t.L.	278.67	2196.00	31.00	99.23	99.78
			0.6	3	0	t.l.	t.L.	285.00	773.33	36.67	96.48	99.78
			0.9	3	2	227.33	t.L.	126.67	450.33	37.67	21.86	65.57
Total				81	29							

Table 6.2: Performance of EXACT_MCFBP Branch-and-Cut algorithm used to solve Model (6.1) on SNDLIB instances

					HEU_MCFBP							
					# opt	time		# lazy	# user	nodes	opt gap	
n	m	d	λ	#	# opt	avg.	max	avg.	avg.	avg.	avg.	max
26	84	15	0.2	9	0	t.l.	t.l.	707.67	11335.89	269.78	92.22	100.00
			0.6	9	1	563.44	t.l.	852.00	199.89	687.56	53.71	80.75
			0.9	9	9	1.44	3.00	45.78	0.00	31.44	0.00	0.00
27	102	16	0.2	6	0	t.l.	t.l.	782.33	126.33	189.83	88.19	95.15
			0.6	6	0	t.l.	t.l.	751.33	52.17	434.50	71.36	85.00
			0.9	6	6	4.17	15.00	61.50	0.00	38.00	0.00	0.00
39	122	23	0.2	9	0	t.l.	t.l.	583.44	26.67	181.56	91.23	99.51
			0.6	9	1	536.22	t.l.	512.56	3.56	224.33	74.49	95.56
			0.9	9	8	71.56	t.l.	98.00	0.00	88.22	5.85	52.63
	172	23	0.2	3	0	t.l.	t.l.	487.33	56.67	87.00	99.23	99.68
			0.6	3	0	t.l.	t.l.	488.33	0.00	105.00	96.48	99.74
			0.9	3	2	230.33	t.l.	257.00	0.00	100.33	24.19	72.58
Total				81	27							

Table 6.3: Performance of HEU_MCFBP Branch-and-Cut algorithm used to solve Model (6.1) on SNDLIB instances

Tables 6.4, 6.5 and 6.6 present the results of the natural formulation (6.1) applied to TELECOM instances, categorized by type: TELECOM_A, TELECOM_B, and TELECOM_C. It is worth noticing that TELECOM_A instances comprise graphs of smaller size (with 71

vertices and approximately 500 arcs) and feature a substantial number of commodities (60). In contrast, the graphs of TELECOM_B are larger (1271 vertices and approximately 2500 arcs) but have fewer commodities (30). Lastly, the TELECOM_C graphs are significantly larger (with 5021 vertices and around 8000 arcs) and have the fewest commodities (10). Similar to previous observations, three values of λ are considered: 0.2, 0.6, and 0.9. The column # reports the total number of instances per row. For each configuration of the Branch-and-Cut algorithm, we report the following values: the number of instances solved to proven optimality (#opt), the average (avg.) and maximum (max) computing time (time) in seconds, the average number of constraints generated to separate integer infeasible points (#lazy), the average number of nodes in the branching tree (nodes), the average and maximum optimality gap value (opt gap). For EXACT_MCFBP and HEU_MCFBP, we also report the average number of constraints generated to separate fractional infeasible points (#user) and the average time required (time user).

First, Tables 6.4, 6.5 and 6.6 support the results of Tables 6.1, 6.2 and 6.3, regarding the computing time in accordance with the target profit value. Indeed, as for SNDLIB instances, we observe that for all configurations of the Branch-and-Cut algorithm, and regardless of the instance type, computing time increases as the value of λ decreases. As an illustrative example, within the INT_MCFBP configuration and for TELECOM_A instances, the average computing time decreases significantly from 37 seconds for $\lambda = 0.2$ to 0.97 seconds for $\lambda = 0.9$. This trend is even more significant for larger instances such as TELECOM_B, where only 5 out of 15 instances were solved to proven optimality within the time limit for $\lambda = 0.2$. In contrast, for $\lambda = 0.9$, all 15 instances were solved to proven optimality, with the maximum computing time not exceeding 58 seconds.

Second, the performance analysis of the three configurations of the Branch-and-Cut (B&C) algorithm indicates that EXACT_MCFBP yields the least favorable results. Specifically, for small-sized instances, INT_MCFBP achieves the optimal solution within a reasonable time (up to 37 seconds for TELECOM_A instances), whereas EXACT_MCFBP requires 167 seconds to solve the same instances, which is more than a fourfold increase. As previously, this is due to the time needed to separate fractional infeasible points by solving the exact separation problem (6.3). For example, it takes 125 seconds to generate 79 #user cuts. On the other hand, using the heuristic algorithm SP_HEU can help reduce the number of fractional integer points to separate (# lazy), for example 62 instead of 67 for TELECOM_A instances.

Finally, we observe that the instances requiring the most time to reach the optimal solution are those of TELECOM_B. Compared to TELECOM_A, the graphs of TELECOM_B instances are larger (having a larger number of vertices and arcs), yet the number of commodities is reduced by half, i.e., 30 commodities instead of 60. Additionally, compared to TELECOM_C, the graphs of TELECOM_B are smaller but have more commodities (30 instead of 10). For TELECOM_A and TELECOM_C instances, all instances have been solved to proven optimality. However, for TELECOM_B with $\lambda = 0.6$, 9 out of 15 instances were optimally solved within the time limit, and with $\lambda = 0.2$, this number decreases to 5. Thus, it can be deduced that the number of commodities in the graph significantly impacts the performance of the model. However, the size of the graph also plays a crucial role in determining the model's performance. As shown in this table, an MCFBP instance consisting of a large graph with a small number of demands and an

MCFBP instance composed of a small graph with a large number of demands can both be solved within a reasonable time (up to 392 seconds with `INT_MCFBP` for instances of `TELECOM_C`). However, the combination of a relatively large graph with a substantial number of commodities (as in `TELECOM_B`) poses a challenge to the model.

			INT_MCFBP						
				time		# lazy	nodes	opt gap	
Type	λ	#	# opt	avg.	max	avg.	avg.	avg.	max
TELECOM_A	0,2	15	15	37,78	271,25	67,27	33,33	0	0
	0,6	15	15	6,54	45,39	16,93	6,27	0	0
	0,9	15	15	0,97	1,65	2,13	0	0	0
TELECOM_B	0,2	15	5	444,88	t.l.	591,2	221,73	25,66	99,56
	0,6	15	9	318,94	t.l.	498,4	268,27	14,15	68,75
	0,9	15	15	13,51	58,6	31,8	11,6	0	0
TELECOM_C	0,2	15	15	40,65	392,25	100,47	91,4	0	0
	0,6	15	15	4,87	22,58	12,6	2,73	0	0
	0,9	15	15	1,21	4,29	2,33	0	0	0
Total		135	119						

Table 6.4: Performance of `INT_MCFBP` Branch-and-Cut algorithm used to solve Model (6.1) on `TELECOM` instances

			EXACT_MCFBP								
			# opt	time		# lazy	# user	time user	nodes	opt gap	
Type	λ	#	# opt	avg.	max	avg.	avg.	avg.	avg.	avg.	max
TELECOM_A	0,2	15	12	167,94	t.l.	70,47	79,33	125,35	26,87	0	0
	0,6	15	13	86,64	t.l.	14,87	45,27	80,39	16,07	0	0
	0,9	15	15	0,96	1,71	2,13	0	0	0	0	0
TELECOM_B	0,2	15	2	t.l.	t.l.	271,8	471,87	505,18	76,33	64,37	99,98
	0,6	15	5	587,15	t.l.	225,67	405,33	449,23	96,13	35,92	95,89
	0,9	15	13	100,09	t.l.	28,93	71,73	85,95	21,4	0	0
TELECOM_C	0,2	15	11	196,5	t.l.	39,73	210,2	167,39	10,8	15,39	99,99
	0,6	15	13	97,81	t.l.	11,4	96,87	81,25	0,73	2,82	42,31
	0,9	15	15	1,82	11,84	2,33	0	0	0	0	0
Total		135	99								

Table 6.5: Performance of `EXACT_MCFBP` Branch-and-Cut algorithm used to solve Model (6.1) on `TELECOM` instances

			HEU_MCFBP								
			# opt	time		# lazy	# user	time user	nodes	opt gap	
Type	λ	#	# opt	avg.	max	avg.	avg.	avg.	avg.	avg.	max
TELECOM_A	0,2	15	15	34,18	217,04	62,93	0,13	0,07	30,4	0	0
	0,6	15	15	6,76	46,34	16,93	0	0,02	6,27	0	0
	0,9	15	15	0,97	1,68	2,13	0	0	0	0	0
TELECOM_B	0,2	15	5	446,1	t.l.	586,8	0	0,93	217,47	30,85	99,56
	0,6	15	9	320,74	t.l.	493,27	0	1,17	263,2	14,5	68,75
	0,9	15	15	13,86	60,42	31,8	0	0,06	11,6	0	0
TELECOM_C	0,2	15	15	41,58	414,57	100,47	0	0,67	91,4	0	0
	0,6	15	15	4,42	23,26	12,6	0	0,02	2,73	0	0
	0,9	15	15	1,11	2,85	2,33	0	0	0	0	0
Total		135	119								

Table 6.6: Performance of HEU_MCFBP Branch-and-Cut algorithm used to solve Model (6.1) on TELECOM instances

In what follows, we conduct a comprehensive performance analysis of the model, examining the impact of graph-structural features, flow-blocker features, and commodity features on its efficiency. This analysis specifically targets instances from the SYNTHETIC class. For solving these instances, our attention is directed towards the INT_MCFBP and HEU_MCFBP configurations of the Branch-and-Cut (Branch-and-Cut) algorithm. Notably, the EXACT_MCFBP configuration has not proven to be efficient. Moreover, as the size of the instance grows, the exact separation problem 6.3 demands significant memory resources, and reaching the optimal solution may also consume considerable time.

Figures 6.3, 6.4, 6.5 and 6.6 give the computing time boxplots of the different Branch-and-Cut algorithms to solve the MCFBP, i.e., INT_MCFBP and HEU_MCFBP. The considered instances belong to the SYNTHETIC SAME class and contain 1200 graphs, grouped by the number of vertices n, the density $d(G)$, value λ for the target flow and the number of commodities d considered. We graphically show the time spent by each configuration of the Branch-and-Cut algorithm through their quartiles. The lines extending vertically from the boxes indicate the variability outside the upper and lower quartiles. Above the upper quartile, the outliers are plotted as individual points. The y-axis is the computing time (in logarithmic scale) and the x-axis represents the group of instances. On the top part of the figure, we report, for each group, the total number of instances solved to proven optimality (#Opt). These boxplots illustrate that, in average, the computing time of HEU_MCFBP is lower than that of INT_MCFBP. Moreover, we notice a discrepancy in the number of instances solved to proven optimality, with HEU_MCFBP achieving a higher success rate, especially when the size of the graph increases. Specifically, for the smallest graphs (with 50 vertices and a density of 0.2), the number of instances solved to proven optimality by INT_MCFBP exceeds that of HEU_MCFBP, with respective counts of 298 and 253 for $n = 50$ and $d(G) = 0.2$. For the same parameters, HEU_MCFBP solves 296 and 252 instances to proven optimality, slightly fewer than INT_MCFBP. However, for larger graphs, this trend reverses. For example, in the case of graphs with 70 vertices,

HEU_MCFBP optimally solves 279 instances, which is 18 more instances than INT_MCFBP.

For all configurations of the algorithms, we observe that the computing time is directly correlated to the graph size, i.e., the number of vertices and the density of the graph. Indeed, these two parameters are directly correlated to the number of variables and constraints of Model (6.1). As an illustrative example, for graphs of 200 vertices, HEU_MCFBP manages to solve only 84 instances out of 300 while for graphs of 50 vertices, only 4 instances remain unsolved. Moreover, for graphs with a density of 0.8, HEU_MCFBP successfully solves 202 instances out of 300 instances, whereas for $d(G)$ equal to 0.2, 252 instances have been solved to proven optimality. We observe a similar behavior when increasing the number of commodities from 10 to 20: the computing time increases, and similarly, the number of instances optimally solved decreases. For example, with INT_MCFBP, the number of instances decreases from 182 for 10 commodities to 179 for 20 commodities and 163 for 30 commodities. Finally, the boxplots demonstrate that the computing time decreases as the target flow increases. In other words, the model is more efficient for large values of the target flow. This is consistent with the constraints of the natural formulation (6.1). Indeed, due to the structure of the target profit inequalities (6.1b), as the value of the target flow Φ increases, the collection of extreme points in $\mathcal{P}_{mcf}$ decreases, leading to fewer constraints in the model.

Figure 6.3: Computing time boxplots of the natural formulation (6.1) for the MCFBP on SYNTHETIC SAME instances, depending on the number of vertices n

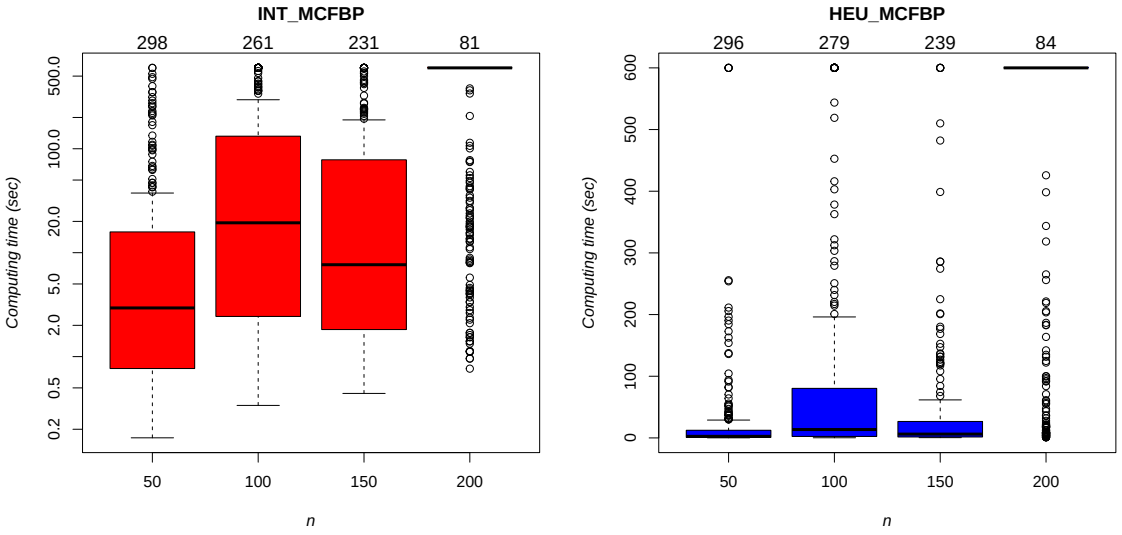


Figure 6.4: Computing time boxplots of the natural formulation (6.1) for the MCFBP on `SYNTHETIC SAME` instances, depending on the density $d(G)$

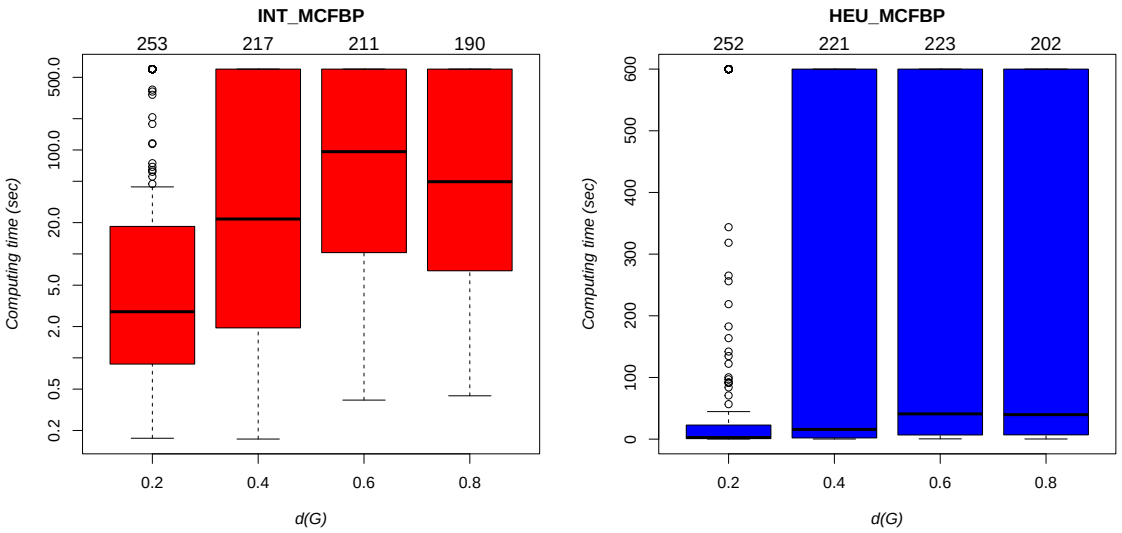


Figure 6.5: Computing time boxplots of the natural formulation (6.1) for the MCFBP on `SYNTHETIC SAME` instances, depending on λ

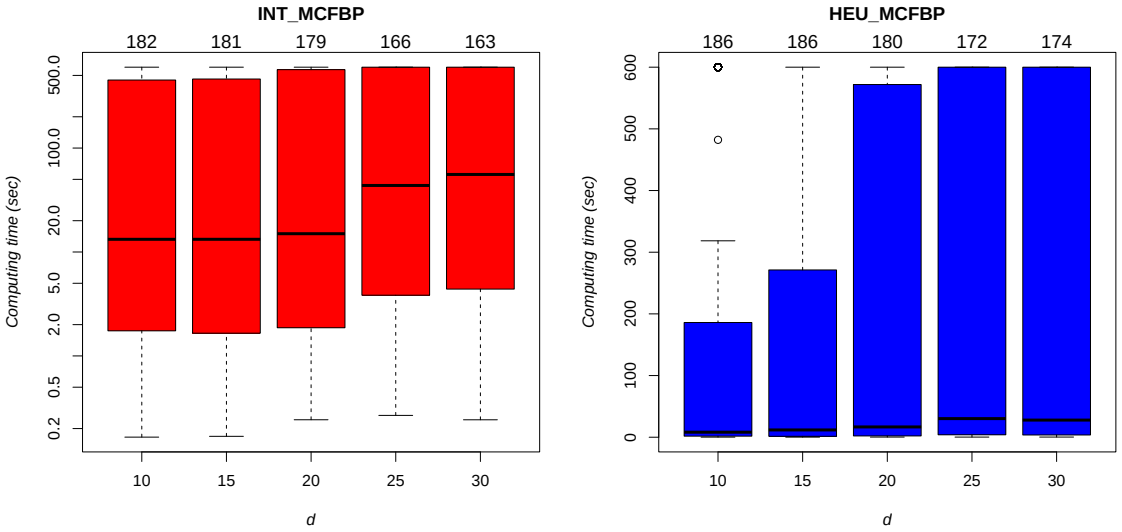
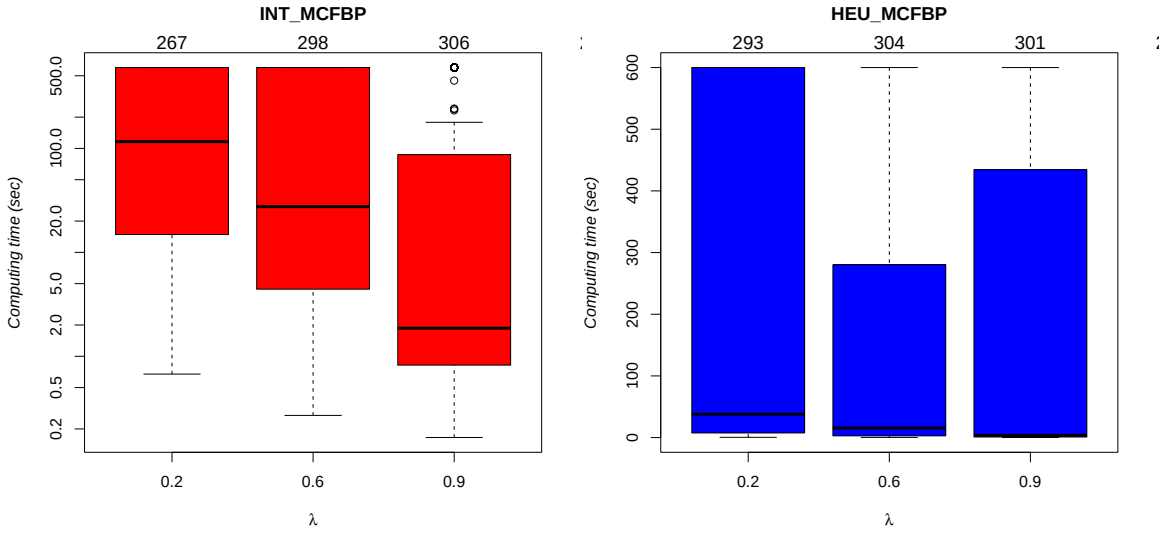


Figure 6.6: Computing time boxplots of the natural formulation (6.1) for the MCFBP on `SYNTHETIC SAME` instances, depending on the number of commodities d



The same observations hold for instances from the `SYNTHETIC_DIFF` instances. However, it is worth noticing that the formulation requires more time to solve when dealing with commodities having different sources and destinations, as shown by the boxplots represented in Figures 6.7, 6.8, 6.9 and 6.10.

Figure 6.7: Computing time boxplots of the natural formulation (6.1) for the MCFBP on `SYNTHETIC DIFF` instances, depending on the number of vertices n

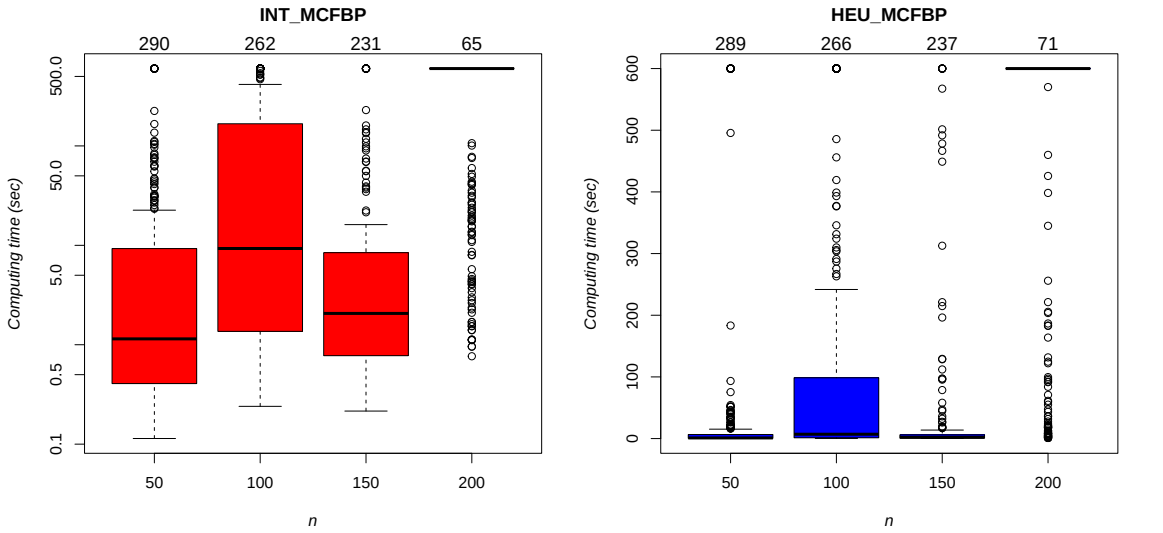


Figure 6.8: Computing time boxplots of the natural formulation (6.1) for the MCFBP on `SYNTHETIC DIFF` instances, depending on the density $d(G)$

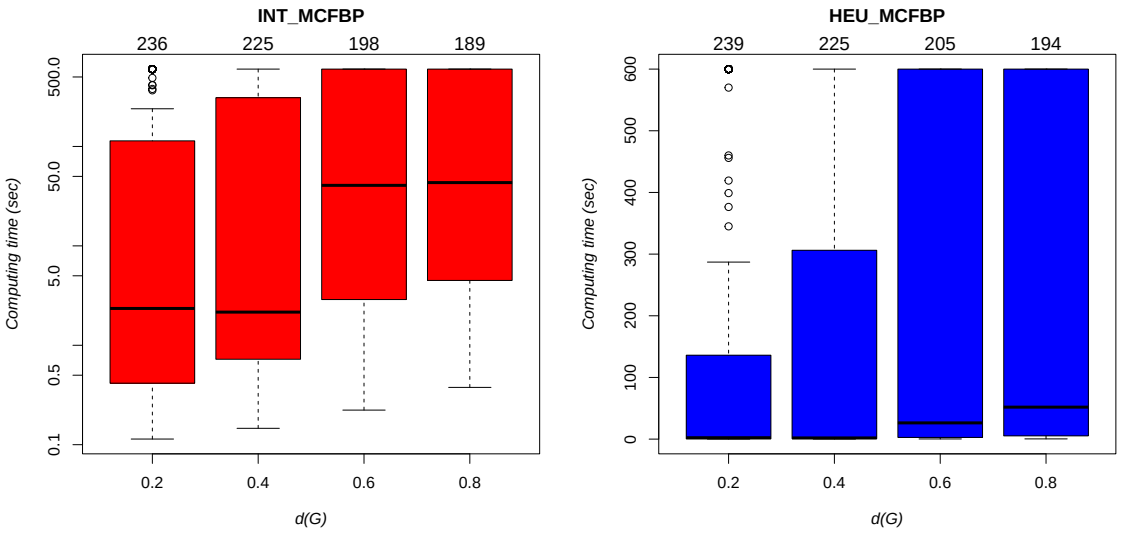


Figure 6.9: Computing time boxplots of the natural formulation (6.1) for the MCFBP on `SYNTHETIC DIFF` instances, depending on λ

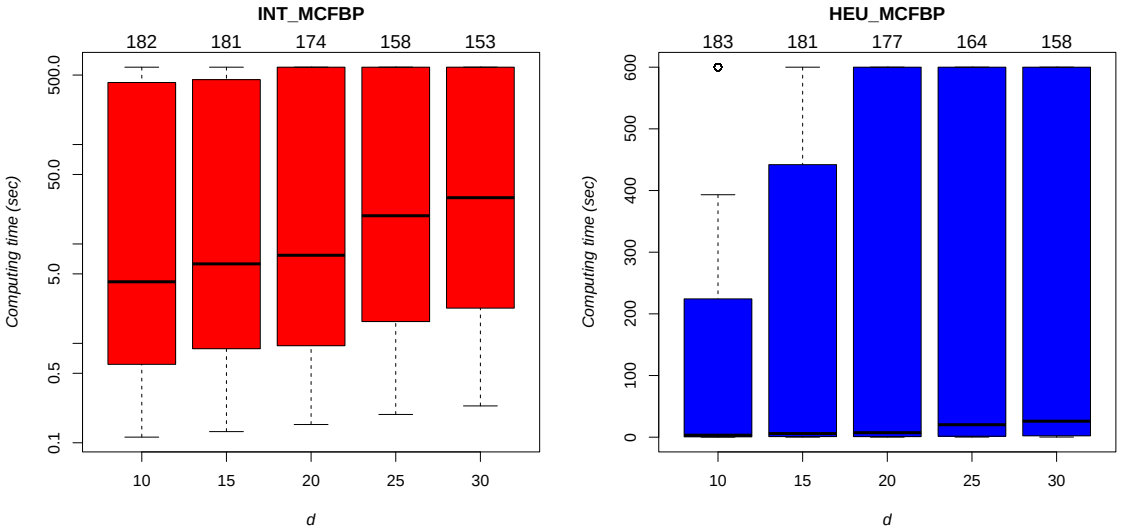
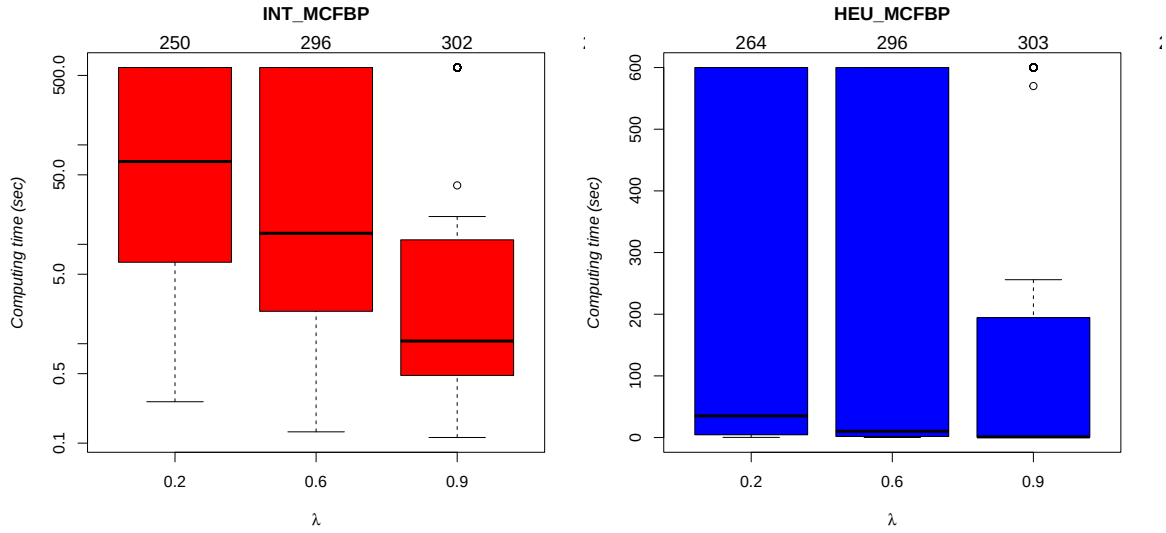


Figure 6.10: Computing time boxplots of the natural formulation (6.1) for the MCFBP on SYNTHETIC DIFF instances, depending on the number of commodities d



6.3.3 Computational performance of the Branch-and-Cut for the UMCFBP

As for the MCFBP, we now present a detailed performance analysis of Model (6.7) to solve the UMCFBP, examining the impact of graph-structural features, flow-blocker features and commodity features on its efficiency.

The natural formulation (6.7) is solved using a Branch-and-Cut algorithm where u-target profit inequalities (6.7b) are separated at every node of the branching tree and added to the RMP. As explained previously, the separation of u-target profit inequalities is $\mathcal{NP}$-hard for fractional and integer RMP solutions (see Proposition 28). Consequently, an efficient separation procedure is crucial. In light of this, we have explored several strategies to separate u-target profit inequalities (6.1b). The first approach involves separating only integer unfeasible points by solving a UMCFP in the non-blocked graph. This configuration of the Branch-and-Cut algorithm is denoted by `INT_UMCFBP`. The second approach, referred to as `INT_CG_UMCFBP` entails the initial heuristic separation of integer infeasible points using the column-generation-based algorithm outlined in Section 6.2.2, followed by the exact resolution of a UMCFP in the non-blocked graph if no cut has been identified. Finally, the third approach, referred to as `CG_HEU_UMCFBP` involves, in addition to `INT_CG_UMCFBP`, the separation of fractional solutions using the shortest-path-based heuristic `U_SP_Heu` presented in Section 6.2.2. It is worth noticing that we did not report the results obtained when solving the exact separation problem (6.9) of u-target profit inequalities to separate fractional RMP solutions. Indeed, this approach has proven to be inefficient when addressing the splittable variant. For the unsplittable variant, separating a single fractional point could significantly increase the computing time.

The three configurations of the Branch-and-Cut algorithm for solving the UMCFBP are compared on `SNDLIB` and `TELECOM` instances.

Table 6.7, 6.8 and 6.9 compare the performance of the three Branch-and-Cut algorithms on SNDLIB instances for three target profit values defined by $\lambda = 0.2$, $\lambda = 0.6$ and $\lambda = 0.9$. Each row in the table corresponds to instances from SNDLIB class grouped by the size, defined by the number of vertices n in the graph, the number of arcs m, the number of commodities d, and the value λ to define the target profit. The column # reports the total number of instances per row. For each configuration of the Branch-and-Cut algorithm, we report the following values: the number of instances solved to proven optimality (#opt), the average (avg.) and maximum (max) computing time (time) in seconds, the average number of constraints generated to separate integer infeasible points (#lazy), the average number of nodes in the branching tree (nodes), the average and maximum optimality gap value (opt gap). The optimality gap represents the percentage discrepancy between the current solution (LB) and the best possible solution (UB) and it is computed as $100 \times \frac{UB-LB}{UB}$. For CG_HEU_UMCFBP, we also report the average number of constraints generated to separate fractional infeasible points (#user).

					INT_UMCFBP						
						time		# lazy	nodes	opt gap	
n	m	d	λ	#	# opt	avg.	max	avg.	avg.	avg.	max
26	84	15	0.2	9	1	587.20	t.l.	661.56	238.44	90.73	98.73
			0.6	9	2	528.88	t.l.	679.33	472.44	46.65	88.11
			0.9	9	9	31.87	142.43	43.33	16.22	0.00	0.00
27	102	16	0.2	6	0	t.l.	t.l.	692.33	201.17	94.87	99.39
			0.6	6	0	t.l.	t.l.	655.67	265.17	85.54	90.61
			0.9	6	6	65.84	117.73	93.33	57.83	0.00	0.00
39	122	23	0.2	9	0	t.l.	t.l.	540.33	122.11	97.52	99.44
			0.6	9	0	t.l.	t.l.	512.11	261.00	80.13	90.70
			0.9	9	9	14.39	36.45	17.78	3.67	0.00	0.00
	172	23	0.2	3	0	t.l.	t.l.	480.33	107.00	99.73	99.80
			0.6	3	0	t.l.	t.l.	433.33	91.33	99.77	99.80
			0.9	3	2	255.48	t.l.	166.67	87.33	18.75	56.25
Total				81	29						

Table 6.7: Performance of INT_UMCFBP Branch-and-Cut algorithm used to solve Model (6.7) on SNDLIB instances

					INT_CG_UMCFBP						
						time		# lazy	nodes	opt gap	
n	m	d	λ	#	# opt	avg.	max	avg.	avg.	avg.	max
26	84	15	0.2	9	2	580.77	t.l.	3745.22	1685.33	78.99	98.17
			0.6	9	4	441.87	t.l.	1677.78	1464.44	24.93	73.86
			0.9	9	9	21.80	85.95	43.33	16.22	0.00	0.00
27	102	16	0.2	6	0	t.l.	t.l.	2335.83	860.83	91.66	96.45
			0.6	6	0	t.l.	t.l.	1736.50	951.17	78.09	87.34
			0.9	6	6	57.86	131.09	115.67	89.83	0.00	0.00
39	122	23	0.2	9	0	t.l.	t.l.	3122.44	1158.89	92.30	94.59
			0.6	9	2	550.35	t.l.	1451.00	975.67	58.80	86.34
			0.9	9	9	15.85	36.27	17.78	3.67	0.00	0.00
	172	23	0.2	3	0	t.l.	t.l.	2529.00	730.33	98.56	99.71
			0.6	3	0	t.l.	t.l.	1475.33	314.00	95.36	97.08
			0.9	3	2	231.44	t.l.	304.67	188.67	0.00	0.00
Total				81	34						

Table 6.8: Performance of `INT_CG_UMCFBP` Branch-and-Cut algorithm used to solve Model (6.7) on `SNDLIB` instances

					CG_HEU_UMCFBP							
						time		# lazy	# user	nodes	opt gap	
n	m	d	λ	#	# opt	avg.	max	avg.	avg.	avg.	avg.	max
26	84	15	0.2	9	3	565.27	t.l.	3572.44	589.33	2602.89	53.78	82.05
			0.6	9	6	400.93	t.l.	1105.78	1895.67	1369.78	18.25	98.00
			0.9	9	9	23.38	57.25	43.33	0.00	16.22	0.00	0.00
27	102	16	0.2	6	0	t.l.	t.l.	2481.17	540.83	1023.33	83.03	92.92
			0.6	6	0	t.l.	t.l.	1743.50	193.33	1115.50	70.38	81.57
			0.9	6	6	55.56	132.01	116.67	0.17	91.50	0.00	0.00
39	122	23	0.2	9	0	t.l.	t.l.	3176.22	324.78	1792.00	81.79	94.30
			0.6	9	2	544.19	t.l.	1518.56	5.11	1106.78	53.20	84.77
			0.9	9	9	18.83	45.21	17.78	0.00	3.67	0.00	0.00
	172	23	0.2	3	0	t.l.	t.l.	2582.67	325.33	680.00	95.68	99.29
			0.6	3	0	t.l.	t.l.	1571.67	0.00	373.33	95.25	97.08
			0.9	3	2	229.96	t.l.	303.00	0.00	190.67	0.00	0.00
Total				81	37							

Table 6.9: Performance of `CG_HEU_UMCFBP` Branch-and-Cut algorithm used to solve Model (6.7) on `SNDLIB` instances

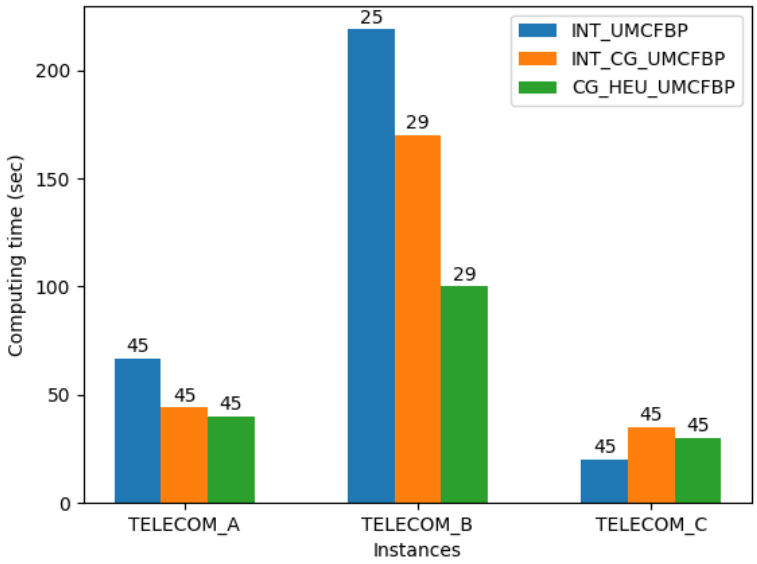
First, our focus is on separating integer infeasible points. For this purpose, we observe that INT_CG_UMCFBP provides better results than INT_MCFBP. Indeed, while INT_MCFBP achieves optimality for 29 out of 81 instances, INT_CG_UMCFBP successfully solves 5 more instances. Additionally, the average and maximum computing times of INT_CG_UMCFBP consistently remain smaller than those of INT_UMCFBP. However, INT_CG_UMCFBP requires the separation of a higher number of integer points (# lazy) and the exploration of more nodes in the branching tree. For example, in the case of the smallest graphs with 26 vertices, 84 arcs, and 15 commodities, with $\lambda = 0.2$, the average number of integer points to be separated stands at 661 when using INT_UMCFBP, but rises to 3745 when employing INT_CG_UMCFBP. Similarly, INT_UMCFBP requires 238 nodes in the branching tree, whereas INT_CG_UMCFBP averages 1685. However, despite this disparity, INT_CG_UMCFBP exhibits superior computing times and overall performance. Moreover, it is worth noticing that the optimality gap (opt gap) consistently remains smaller for INT_CG_UMCFBP, indicating that even when it does not achieve the optimum, INT_CG_UMCFBP provides a better solution compared to INT_UMCFBP, i.e., a solution closer to optimality.

We now assess the gain of separating fractional solutions using the shortest-path-based heuristic presented in Section 6.2.2. We observe that separating fractional infeasible points results in a reduction in the number of integer points that need to be separated, thereby improving the overall performance. This is evidenced by shorter computing times for CG_HEU_UMCFBP, a higher number of instances solved to proven optimality (37), and smaller values for the optimality gap.

Therefore, we can conclude that CG_HEU_UMCFBP has proven to be the most efficient configuration of the Branch-and-Cut algorithm for solving the natural formulation (6.7) for the UMCFBP, particularly on SNDLIB instances. To validate this finding, we will conduct a similar analysis on TELECOM instances.

In Figure 6.3.3, we present a comparative analysis of the three algorithms for solving the UMCFBP, i.e., INT_UMCFBP, IN_CG_UMCFBP and CG_HEU_UMCFBP on TELECOM instances. Specifically, Figure 6.3.3 illustrates a bar chart depicting the average computing time of each algorithm on TELECOM_A, TELECOM_B and TELECOM_C instances. Each category of instances is represented by three adjacent bars, corresponding to an algorithm. Additionnaly, above each bar, we report the number of instances solved to proven optimality. This figure primarily highlights the effectiveness of employing the heuristic approach to separate integer unfeasible points, thereby enhancing the algorithm's overall performance.

Figure 6.11: Performance comparison of the natural formulation (6.7) for the UMCFBP on TELECOM instances



It is worth noticing that the natural formulation becomes more challenging to solve when considering unsplittable flows, especially large graphs. These observations can be explained mostly by the $\mathcal{NP}$-hardness of the UMCFP, resulting in a longer separation process for integer infeasible points. However, our experiments indicate that the heuristic employed to separate fractional infeasible points performs more effectively when applied to unsplittable flows.

6.3.4 Comparison of the effectiveness of the natural formulation and the state-of-the-art technique

In this section, we are interested in comparing performance of the Branch-and-Cut algorithms presented against the direct use of an online-accessible solver designed for bilevel programs, which corresponds to the current state-of-the-art approach to solving the MCFBP and the UMCFBP.

This exact solver designed specifically for mixed-integer bilevel linear programs (MIBLPs) has been introduced in Chapter 4 to solve the MFBP. As mentioned in this chapter, it is available online [2] and it is associated with the work of Fischetti and Ljubic [2017]. In this study, we employ the same configuration of the solver as the one presented for the MFBP, namely the MIX++ setting. Finally, in accordance with the tests made for the natural formulations of the MCFBP and the UMCFBP, the solver is used in a single-thread mode. This resolution method is denoted by BISOLVER.

For this study, we consider instances from the SNDLIB and SYNTHETIC classes, and for each instance, we set a time limit of 600 seconds.

[2] https://msinnl.github.io/pages/bilevel.html

BISOLVER technique for the MCFBP

In this section, we focus on performance of the BISOLVER technique to solve the MCFBP.

First, experiments indicate that when using BISOLVER, no instances from the SNDLIB class were solved to proven optimality within the time limit, regardless of the values of λ. On the contrary, the best configuration of the Branch-and-Cut algorithm for the natural formulation (6.1), which corresponds to EXACT_MCFBP for these instances, manages to solve 29 instances out of 81 within the time limit.

We now analyze the performance of BISOLVER on instances drawn from the SYNTHETIC SAME and SYNTHETIC DIFF classes. This investigation aims to evaluate the effectiveness of the state-of-the-art technique in addressing the MCFBP's challenges and to compare it with the natural formulation (6.1) proposed in this thesis for the same purpose.

Figure 6.3.4 and Figure 6.3.4 present a comparative analysis of two methodologies for solving the MCFBP. Figure 6.3.4 focuses on instances from the SYNTHETIC SAME class, while Figure 6.3.4 focuses on instances from the SYNTHETIC DIFF class. Specifically, these two figures compare the Branch-and-Cut algorithm HEU_MCFBP for solving the natural formulation (6.1) against the direct use of a bilevel solver, referred to as BISOLVER. The figures illustrate their respective average computing time relative to the number of vertices n in the graph. The x-axis represents the values of n and for each value, the average computing time is displayed in the y-axis. The blue curve represents the performance of HEU_MCFBP, while the orange curve depicts that of BISOLVER. Additionally, we report on each point of the curve the total number of instances solved to proven optimality. It is worth noticing that our analysis encompasses 9 groups of instances categorized by the number of vertices, with each group consisting of 180 graphs.

Figure 6.3.4 shows that the computing time of BISOLVER consistently exceeds that of HEU_MCFBP. Specifically, for all instances, the computing time of HEU_MCFBP remains below 60 seconds, whereas the computing time of BILEVEL fluctuates between 100 and 350 seconds. Furthermore, it is worth noticing that, except for graphs with 70 vertices where two instances remain unsolved, HEU_MCFBP successfully solves all instances to proven optimality. Conversely, for the smallest graphs (20 vertices), the bilevel solver achieves optimality only for 140 out of 180 instances. This number decreases to 98 instances for graphs with 100 vertices.

Figure 6.3.4 exhibits similarities to Figure 6.3.4, with the performance of the bilevel solver remaining the same on instances where commodities have different sources and destinations. The primary distinction between the two figures lies in the performance of the Branch-and-Cut algorithm, i.e., HEU_MCFBP. In Figure 6.3.4, the number of instances optimally solved decreases to 173 out of 180 for graphs with 70 vertices, and the computing time increases to approximately 25 seconds for graphs with 100 vertices. Nonetheless, the performance of HEU_MCFBP remains significantly superior to that of BISOLVER.

Figure 6.12: Performance comparison between the bilevel solver and the natural formulation (6.1) for the MCFBP, on `SYNTHETIC SAME` instances

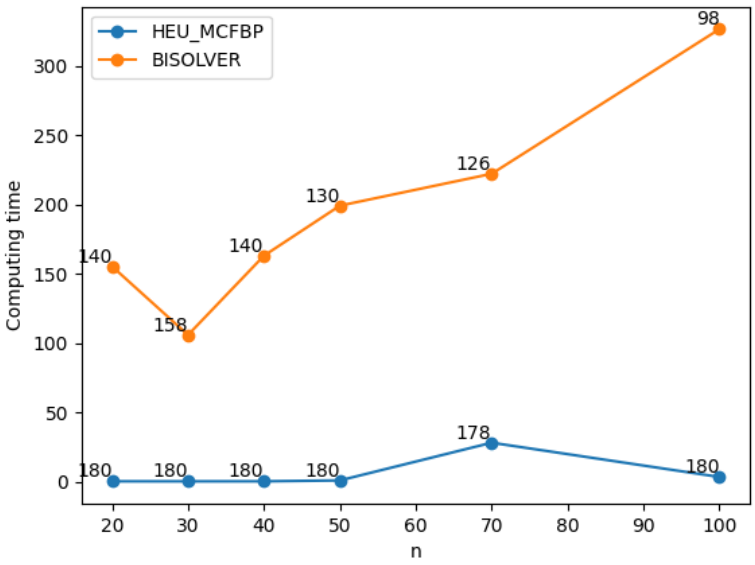
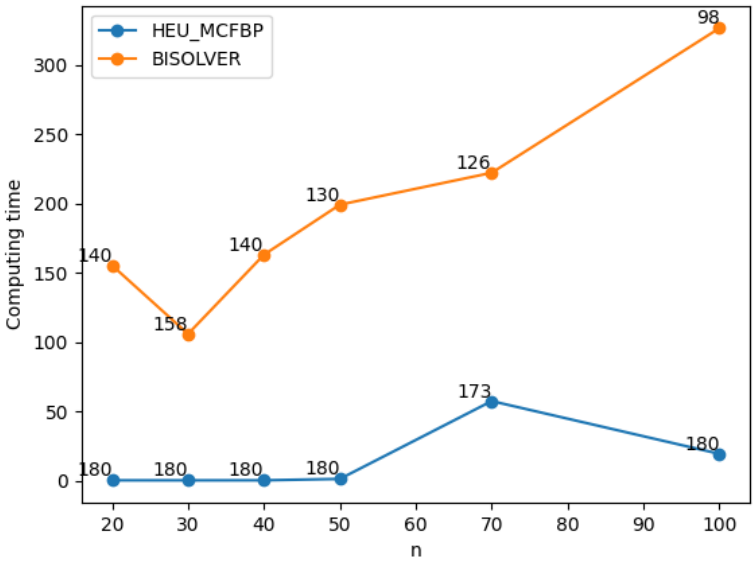


Figure 6.13: Performance comparison between the bilevel solver and the natural formulation (6.1) for the MCFBP, on `SYNTHETIC DIFF` instances



`BISOLVER` technique for the UMCFBP

We are now interested in evaluating performance of the bilevel solver, i.e., `BISOLVER` for solving the UMCFBP.

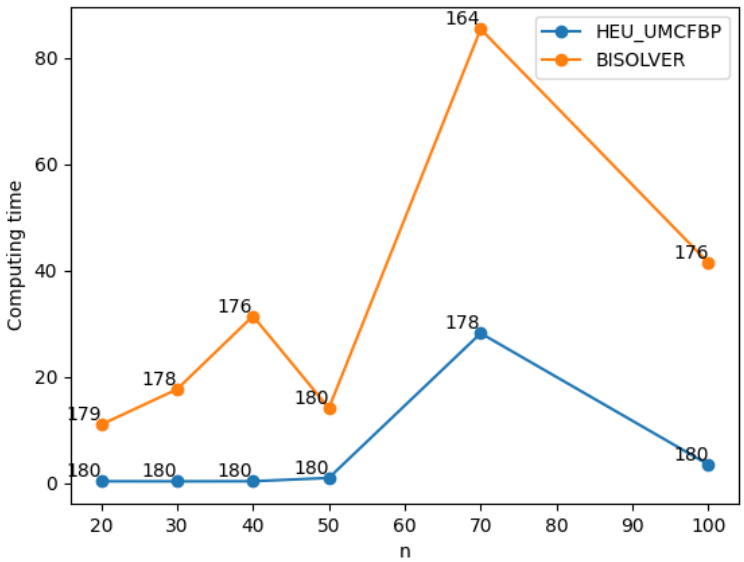
As for the MCFBP, experiments have revealed that using the bilevel solver to address the UMCFBP fails to achieve optimality for instances from the `SNDLIB` class within the time limit. We are currently investigating the performance of `BISOLVER` on instances

from the SYNTHETIC DIFF class. Given the results observed in the previous section for instances with both identical and different sources, we now focus only on the most general instances, featuring distinct sources and destinations.

As previously, Figure 6.3.4 presents a comparative analysis of two methodologies for solving the UMCFBP, focusing on instances from the SYNTHETIC DIFF class. This figure compares the Branch-and-Cut algorithm CG_HEU_UMCFBP (also referred to as HEU_UMCFBP)for solving the natural formulation (6.7) against the direct use of a bilevel solver, referred to as BISOLVER. As in Figure 6.3.4, the x-axis represents the number of vertices n in the graph and the y-axis represents the average computing time. The blue curve represents the performance of CG_HEU_UMCFBP, while the orange curve depicts that of BISOLVER. Additionally, we report on each point of the curve the total number of instances solved to proven optimality. The set of instances is composed by 9 groups containing 180 graphs with the same number of vertices.

The computing time of BISOLVER consistently exceeds that of CG_HEU_UMCFBP. For small instances, ranging from 20 to 50 vertices, the disparity between the two techniques is relatively small. While CG_HEU_UMCFBP provides the optimal solution in a few seconds, the computing time of BISOLVER fluctuates between 15 and 35 seconds, which is comparatively reasonable. However, even for these sizes of graphs, the bilevel solver fails to reach the optimal solution for some instances. For example, for graphs with 20, 30, and 40 vertices, 1, 2, and 4 instances, respectively, have not been solved to proven optimality within the time limit. For larger instances, such as graphs with 70 vertices, the difference between the two methods becomes more evident. For example, the computing time of CG_HEU_UMCFBP approaches 35 seconds, whereas the computing time of BISOLVER exceeds 80 seconds (more than double), with 16 instances remaining unsolved.

Figure 6.14: Comparison between the bilevel solver and the natural formulation (6.7) the UMCFBP, on SYNTHETIC DIFF instances



It is worth noticing that the bilevel solver demonstrates better performance in terms

of computing time for the UMCFBP compared to the MCFBP. This distinction is noticeable when examining the average computing time on the y-axis in figures 6.3.4 and 6.3.4 for the MCFBP, which ranges between 0 and 300 seconds. In contrast, in Figure 6.3.4 for the UMCFBP, the maximum computing time achieved by `BISOLVER` is approximately 90 seconds. This discrepancy arises from the nature of the cuts introduced by the bilevel solver, which are better suited to bilevel models where variables of the follower problem are integers.

6.4 Concluding remarks

In this chapter, we present Branch-and-Cut algorithms designed to address the MCFBP and the UMCFBP. For each variant, we delve into the separation problem concerning the proposed exponential family of constraints. Subsequently, we introduce various approaches to separate these inequalities. For fractional RMP solutions, we propose a heuristic based on successive computations of shortest paths. Additionally, for the UMCFBP, we introduce a column-generation-based algorithm to separate integer infeasible points. To evaluate the effectiveness of the different approaches and assess performance of the Branch-and-Cut algorithm, we conduct an extensive computational study on a set of synthetic and real-world instances. This study highlights the efficiency of the described heuristics, especially for large graphs.

Chapter 7

The multicommodity flow blocker problem: vertex variant

Contents

In the previous chapters, our work centered on the blocker variant of the maximum flow problem and the multicommodity flow problem, whose objective is to destruct arcs from the graph in order to deteriorate the maximum flow and the multicommodity flow, respectively. The current chapter delves into a novel blocker variant, that aims to remove vertices instead of arcs. This chapter specifically targets the vertex blocker variant of the unsplittable multicommodity flow problem, which is called the unsplittable multi-commodity flow vertex blocker problem, denoted by V-UMCFBP. After providing a formal description of the problem, we establish a connection between solutions of the arc blocker variant and solutions of the vertex blocker variant using a graph transformation. We then introduce an Integer Linear Programming (ILP) formulation to address this problem. This formulation is a natural formulation featuring an exponential number of constraints and it is adapted from the one designed for the arc blocker variant of the unsplittable multicommodity flow problem, i.e., the UMCFBP. Following this, we propose a polyhedral analysis of this model offering insights into the structure of the polytope and its solutions. As the formulation proposed uses only design variables, it is solved using a Branch-and-Cut algorithm. Finally, we present some experiments conducted to assess the performance of the natural formulation for the vertex blocker variant.

7.1 The multicommodity flow vertex blocker problem

7.1.1 Definition

Notations

Let us recall the notations introduced in the previous chapter.

We consider a directed graph $G = (V, A)$ with $m = |A|$ arcs and $n = |V|$ vertices. Each arc $a \in A$ has a capacity $c_a \in \mathbb{Z}_+$ and a flow cost $p_a \in \mathbb{Z}_+$, corresponding to the cost for routing one unit of flow. We consider a set of commodities K where a commodity $k = (s_k, t_k, b_k, \Gamma_k)$ is defined by a source $s_k \in V$, a destination $t_k \in V$, a bandwidth $b_k \in \mathbb{Z}_+^*$ and a reward $\Gamma_k \in \mathbb{Z}_+^*$.

For the vertex variant of the multicommodity flow blocker problem, we assign to each vertex $v \in V$, a blocker cost $b_v \in \mathbb{Z}_+$. In this way, the V-UMCFBP consists in finding a minimum-cost subset of vertices to be blocked in such a way that the maximum profit of the UMCF remaining in the graph is no larger than the *target profit* $\Phi \in \mathbb{Z}_+$.

In the next section, we provide a detailed example of a V-UMCFBP solution.

Graphical illustration

We illustrate in this section the features of optimal UMCFP and V-UMCFBP solutions thanks to an example graph with 8 vertices and 12 arcs shown in Figures 7.1 and 7.2. This example includes two commodities $k_1 = (s_1, t_1, d_1 = 8, r_1 = 6)$ and $k_2 = (s_2, t_2, d_2 = 10, r_2 = 35)$. In Figure 7.1, we report on each arc two values separated by the symbol ";": the first one, in brackets, is the flow cost, and the second one, is the capacity of the arc. The curved arcs illustrate the solution of the UMCFP. Red ones indicate the flow routed for commodity k_1 while blue ones represent the flow for commodity k_2. Above these curved arcs, we display the corresponding flow quantities in red or blue. By definition, for every commodity, the demand is routed through a path; $\{s_1, v_1, v_3, t_1\}$ for commodity k_1 and $\{s_2, v_2, v_4, v_3, t_2\}$ for commodity k_2. In this example, the two commodities are satisfied. The total routing profit of the UMCF is equal to 28.

The same graph is then used to illustrate an optimal V-UMCFBP solution in Figure 7.2. We report on each vertex its blocker cost in bold. We consider a target profit $\Phi = 10$ ($\approx 64\%$ of the maximum profit in G). In the optimal V-UMCFBP solution shown, a single vertex, namely v_2, is blocked with a total blocker cost of 3. The blocked vertex is visually marked with a dashed outline. The UMCF remaining in the graph achieves a profit of 28 satisfying the condition $28 \leq \Phi = 10$. It is worth noticing that in this UMCF, only one commodity is satisfied, specifically, commodity k_1.

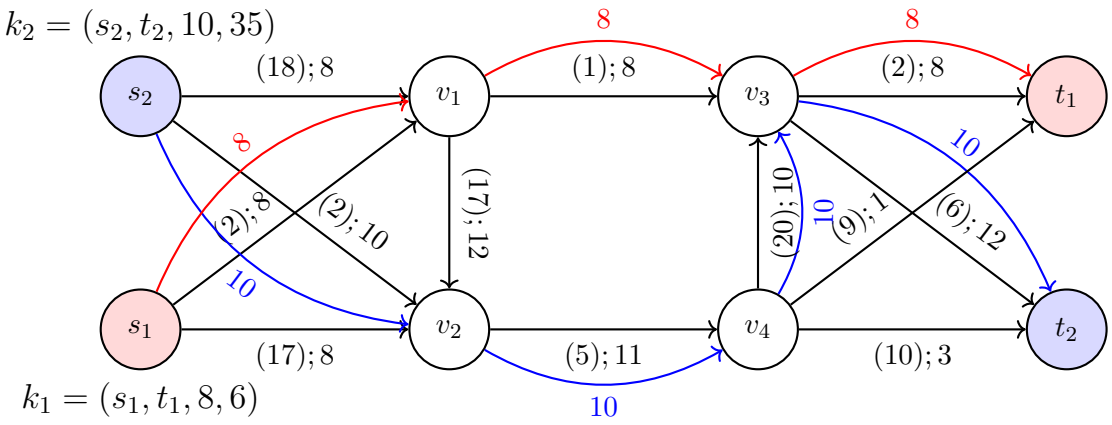


Figure 7.1: An optimal UMCFP solution

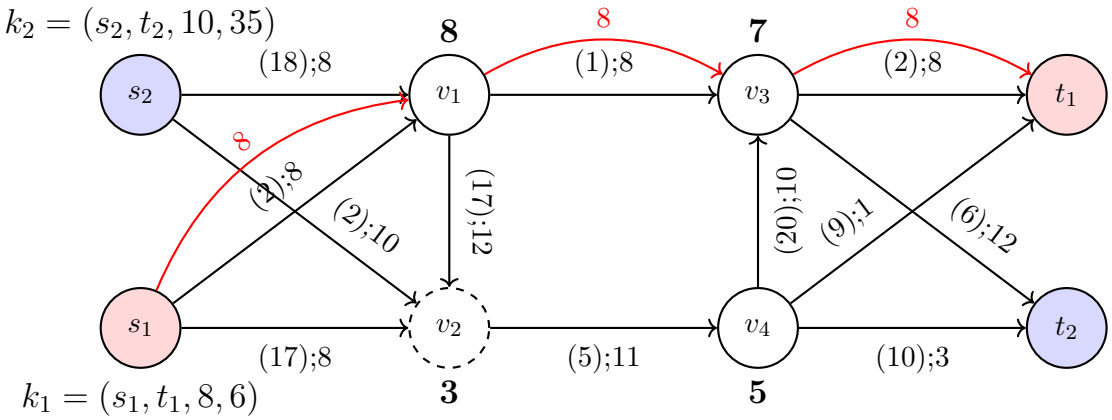


Figure 7.2: An optimal V-UMCFBP solution with a target profit $\Phi = 10$

7.1.2 Relationship between vertex and arc blocker variants

In this section, we demonstrate the equivalence between solving the vertex-blocker variant and the arc-blocker variant of the unsplittable multicommodity flow problem (UMCFP).

Let $G = (V, A)$ be a directed graph characterized by a capacity $c_a \in \mathbb{Z}_+$, a flow cost $p_a \in \mathbb{Z}_+$ and a blocker cost $r_a \in \mathbb{Z}_+$.

We introduce an arc-augmented graph, denoted as $G^* = (V^*, A^*)$, defined such that $V \subseteq V^*$, with $|V^*| = |V| + |A|$ and the number of arcs is doubled, resulting in $|A^*| = 2 \times |A|$, thereby expanding the arc set from the original graph G. The construction process of G^* from G involves the following steps: For each arc $(u, v) \in A$, we create an associated arc-vertex, denoted by $v_{(u,v)} \in V^*$. The original arc (u, v) is then decomposed into two new arcs: $(u, v_{(u,v)}) \in A^*$ and $(v_{(u,v)}, v) \in A^*$. These arcs are assigned the same capacity as the original arc (u, v) and flow costs; the flow cost for the arc $(u, v_{(u,v)})$ is equal to the flow cost of the original arc (u, v), whereas the flow cost of the arc $(v_{(u,v)}, v)$ is assigned a value of 0. Moreover, each newly introduced arc-vertex $v_{(u,v)} \in V^* \setminus V$ is assigned a blocker cost that mirrors the blocker cost associated with the original arc $(u, v) \in A$. On the other hand, each vertex $v \in V$ is attributed an infinite blocker cost.

The figures provided below depict the graph transformation described in this section. Figure 7.3 illustrates a simple graph $G = (V, A)$ having 6 vertices and 5 arcs. For each arc, three distinct values are presented: the blocker cost, highlighted in bold at the top, and below it, two other values separated by a semicolon ";". The first of these values, enclosed in brackets, is the flow cost, and the second is the capacity of the arc. Following this, Figure 7.4 illustrates the arc-augmented graph $G^* = (V^*, A^*)$, which consists of 11 vertices and 10 arcs. Displayed on each arc are the flow cost and the capacity, while the blocker cost is denoted in bold on each vertex.

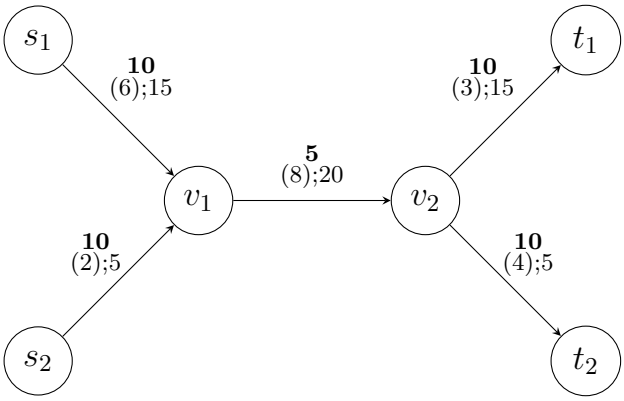


Figure 7.3: A graph G with 6 vertices and 5 arcs.

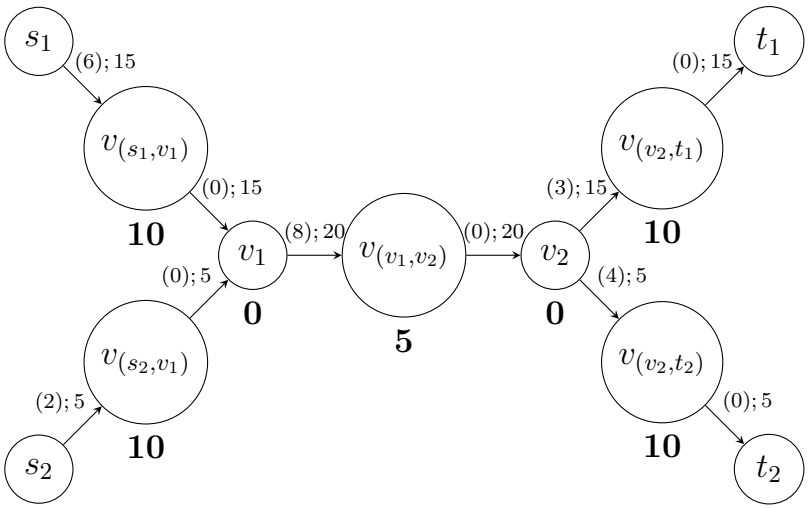


Figure 7.4: The augmented graph G^* with 11 vertices and 5 arcs.

We now introduce the vertex-augmented graph $G^{**} = (V^{**}, A^{**})$, where $V \subseteq V^{**}$ and $A \subseteq A^{**}$. This graph is constructed from the original graph G as follows. For each vertex $v \in V$, we define one vertex $V^{**} \in V^{**}$ such that $(v, V^{**}) \in A^{**}$ with $c_{(v,V^{**})} = max\{c_{(u,v)} : (u, v) \in A\}$, $\omega_{(v,V^{**})} = 0$ and $b_{(v,V^{**})} = b_v$. This arc (v, V^{**}) is called *node-arc* associated to vertex v. Each arc $(v, u) \in A$ is replaced in the vertex-augmented graph by an arc $(V^{**}, u) \in A^{**}$ with $c_{(V^{**},u)} = c_{(v,u)}$, $p_{(V^{**},u)} = p_{(v,u)}$ and $r_{(V^{**},u)} = M$, where M is a large integer value. Moreover, every arc $(u, v) \in A^{**}$ is associated with a blocker cost $r_{(v,u)} = M$.

Thanks to these graph transformations, we prove that there exists an equivalence between the arc-blocker variant of the UMCFP, i.e., the UMCFBP, and the vertex-blocker

variant of the UMCFP, i.e., the V-UMCFBP. More precisely, the next proposition shows that an optimal solution of the UMCFBP solved in G can be obtained by solving the vertex-blocker variant of the UMCFP in the arc-augmented graph G^*.

Proposition 29. *An optimal solution of the UMCFBP in $G = (V, A)$ can be found by solving a V-UMCFBP in $G^* = (V^*, A^*)$.*

Proof. By construction, it is clear that the maximum profit of a UMCF in G is equal to the maximum profit of a UMCF in G^*. In other words, the optimal solution value of a UMCFP solved in G is equal to the optimal solution value of the UMCFP solved in G^*. Moreover, the removal of an arc-vertex in G^* produces the same reduction of maximum profit as the removal of the associated arc in G. Accordingly, when considering a set of blocked vertices $\tilde{V} \subseteq (V^* \setminus V)$ that constitute a feasible solution for the V-UMCFBP solved in G^*, these vertices are associated with a set of blocked arcs $\tilde{A}$ in G.

We first prove that $\tilde{A}$ is a feasible solution for the UMCFBP solved in the graph G. We recall that $\tilde{A}$ is a collection of arcs that are associated with the set of vertices-arcs $\tilde{V}$, which represents a feasible solution for the vertex blocker variant, i.e., the V-UMCFBP solved in G^*. Accordingly, the removal of the arcs $\tilde{A}$ in the graph G ensures that the profit of the UMCF remaining in G does not exceed the target profit value Φ. This guarantees that $\tilde{A}$ is a feasible solution for the UMCFBP solved in G. It is worth noticing that the two solutions, $\tilde{V}$ and $\tilde{A}$, have the same objective value. Indeed, every arc-vertex associated with an arc has a blocker cost equal to the blocker cost of that arc.

We now prove that, if $\tilde{V}$ is an optimal solution for the V-UMCFBP in G^*, then $\tilde{A}$ is also an optimal solution for the UMCFBP in G. By contradiction, suppose that this is not the case, then there exists a set of blocked arcs $\tilde{\tilde{A}}$ in A, that is a feasible solution for the UMCFBP solved in G and such that $\sum_{a \in \tilde{\tilde{A}}} r_a < \sum_{a \in \tilde{A}} r_a$. However, since all vertices $v \in V$, i.e., vertices that are not arc-vertices in V^* have infinite blocker costs, it follows that $\tilde{\tilde{A}}$ is associated with a set of arc-vertices in G^*. This set of arc-vertices forms a feasible solution for the V-UMCFBP solved in the augmented graph G^*. Moreover, it exhibits a lower solution value compared to $\tilde{V}$. However, this contradicts the fact that $\tilde{V}$ is an optimal solution.

□

We can use a similar reasoning to prove that an optimal solution of the V-UMCFBP solved in G can be obtained by solving the arc-blocker variant of the UMCFP in the vertex-augmented graph G^{**}.

Proposition 30. *An optimal solution of the V-UMCFBP in $G = (V, A)$ can be found by solving a UMCFBP in $G^{**} = (V^{**}, A^{**})$.*

Proof. By construction, it is clear that the maximum profit of a UMCF in G is equal to the maximum profit of a UMCF in G^{**}. In other words, the optimal solution value of a UMCFP solved in G is equal to the optimal solution value of the UMCFP solved in G^{**}. Moreover, the removal of a node-arc in G^{**} produces the same reduction of maximum profit as the removal of the associated vertex in G. Accordingly, when considering a set

of blocked vertices $\tilde{V} \subseteq V$ that constitute a feasible solution for the V-UMCFBP solved in G, these vertices are associated with a set of blocked node-arcs $\tilde{A^{**}} \subseteq A^{**}$ in G^{**}.

We first prove that $\tilde{A^{**}}$ is a feasible solution for the UMCFBP solved in the augmented graph G^{**}. We recall that $\tilde{A^{**}}$ is a collection of node-arcs that are associated with the set of vertices $\tilde{V}$, which represents a feasible solution for the V-UMCFBP solved in G. Accordingly, the removal of the arcs $\tilde{A^{**}}$ in the vertex-augmented graph G^{**} ensures that the profit of the UMCF remaining in G^{**} does not exceed the target profit value Φ. This guarantees that $\tilde{A^{**}}$ is a feasible solution for the UMCFBP solved in G^{**}. It is worth noticing that the two solutions, $\tilde{V}$ and $\tilde{A^{**}}$, have the same objective value. Indeed, every node-arc associated with a vertex has a blocker cost equal to the blocker cost of that vertex.

We now prove that, if $\tilde{V}$ is an optimal solution for the V-UMCFBP in G, then $\tilde{A^{**}}$ is also an optimal solution for the arc-blocker, i.e., the UMCFBP in G^{**}. By contradiction, suppose that this is not the case, then there exists a set of blocked arcs $\tilde{\tilde{A}}^{**}$ in A^{**}, that is a feasible solution for the UMCFBP solved in G^{**} and such that $\sum_{a \in \tilde{\tilde{A}}^{**}} r_a < \sum_{a \in \tilde{A^{**}}} r_a$. However, since all arcs $a \in \setminus A$, i.e., arcs that are not node-arcs, have a large blocker cost value equal to M, it follows that $\tilde{\tilde{A}}^{**}$ is a set of node-arcs associated with a set of vertices $\tilde{\tilde{V}} \subseteq V$ in G. This set of vertices forms a feasible solution for the V-UMCFBP solved in G. Moreover, it exhibits a lower solution value compared to $\tilde{V}$. However, this contradicts the fact that $\tilde{V}$ is an optimal solution. Which ends the proof.

□

7.1.3 An ILP formulation for the V-UMCFBP

In this section, we present a model designed to tackle the V-UMCFBP. This model features an exponential family of constraints. As for the natural formulation introduced in Chapter 5 to solve the arc-blocker variant of the UMCFP, namely the UMCFBP, this model is a set-covering type formulation. However, it is explicitly designed to tackle the vertex-blocker variant of the UMCFP. As the natural formulations introduced in the previous chapters, the aim is to provide a concise formulation that can be easily tailored to handle different variants of the problem.

Given a directed graph $G = (V, A)$ and a set K of d commodities, let $\boldsymbol{y} \in \mathcal{P}_{umcf}$ be a vector associated with a UMCF of profit value greater than Φ in graph G, and let $\mathcal{V}_S(\boldsymbol{y}) \subseteq V$ be the subset of vertices in the MCF, i.e.,

$$\mathcal{V}_S(\boldsymbol{y}) = \{u, v \in V : y_{k,(u,v)} > 0, \forall k \in K, \forall (u,v) \in A\}.$$

Let us introduce a vector $\mathbf{x} \in \{0,1\}^n$ of m binary variables, called *vertex blocker variables*. Each variable x_v is associated with a vertex $v \in V$ and it takes value 1 if and only if the vertex v is blocked, i.e., removed from the graph G. We denote by $\mathcal{V}(\mathbf{x})$ the set of blocked vertices induced by a binary realization of the vertex blocker vector $\mathbf{x}$. In other words, $\mathcal{V}(\mathbf{x})$ is the set of vertices $v \in V$ with x_v equal to 1. Suppose that $\mathcal{V}(\mathbf{x})$ and $\mathcal{V}_S(\boldsymbol{y})$ satisfies the following inequality:

$$|\mathcal{V}(\mathbf{x}) \cap \mathcal{V}_S(\boldsymbol{y})| \geq 1, \quad \forall\, \boldsymbol{y} \in ext(\mathcal{P}_{umcf}), \tag{7.1}$$

If so, then $\mathbf{x}$ is a feasible solution for the V-UMCFBP. Indeed, the inequality (7.1) guarantees that no UMCF with a profit greater than Φ remains in the graph after the removal of the blocked vertices. Precisely, the existence of such a UMCF would entail the removal of a vertex, resulting in an obstruction of the routing process.

Based on this reasoning, a valid ILP formulation for the V-UMCFBP reads as follows:

$$\min_{\mathbf{x}\in\{0,1\}^n} \quad \sum_{v\in V} b_v x_v \tag{7.2a}$$

$$\sum_{v\in\mathcal{V}_S(\boldsymbol{y})} x_v \geq 1 \qquad \forall\, \boldsymbol{y} \in ext(\mathcal{P}_{umcf}) \tag{7.2b}$$

The objective function (7.2a) minimizes the total blocker cost. Constraints (7.2b), denoted as V-target profit constraints, are derived from Inequality (7.1). These constraints ensure that at least one vertex from every UMCF with a total profit greater than Φ is blocked. This ILP model is called *natural formulation* since it features only the natural variables associated with the arcs.

A binary realisation $\mathbf{x} \in \{0,1\}^n$ of the vertex blocker variables is called a *vertex blocker policy* and it generates a *vertex non-blocked graph* $\mathcal{G}^V(\mathbf{x}) = \big(\mathcal{V}_{NB}(\mathbf{x}), \mathcal{A}_{NB}(\mathbf{x})\big)$, i.e., the graph with the non-blocked vertices $v \in V$ with $x_v = 0$ (denoted $\mathcal{V}_{NB}(\mathbf{x})$). The set of arcs $\mathcal{A}_{NB}(\mathbf{x})$ contains the arcs remaining in G after removal of the blocked vertices, i.e., $\mathcal{A}_{NB}(\mathbf{x}) = \{(u,v) \in A : x_u = x_v = 0\}$.

The V-target profit inequalities (7.2b) are in exponential number. Accordingly, to solve the natural formulation (7.2), one needs to implement a Branch-and-Cut (B&C) algorithm where V-target profit inequalities are separated in the nodes of the branching tree for integer and fractional solutions. This exact algorithm requires defining a *relaxed master problem* (RMP) where the binary variables are replaced with continuous variables taking values between 0 and 1. In the initialization phase, only a subset of constraints are included in the RMP. To check that RMP solutions respect all the V-target profit inequalities or to determine one or more violated constraints which are then added to the RMP, we propose several separation procedures that will be described subsequently.

7.2 Polyhedral analysis

This section presents a polyhedral analysis of the natural formulation with V-target profit inequalities (see Model (7.2)), which was introduced in the previous section to address the V-UMCFBP.

We make the same assumption as the one made for the arc-blocker variant (see Chapter 5). More precisely, we suppose that there is no arc (s_k, t_k) for all commodities $k \in K$. Moreover, it is worth noticing that this polyhedral study can be adapted from the arc-variant through the graph transformation presented in Section 7.1.2.

Given a directed graph $G = (V, A)$ and a set of commodities K, let $P^V(G, K)$ be the convex hull of the solutions of Formulation (7.2), that is:

$$P^V(G,K) = conv(\{\mathbf{x} \in \{0,1\}^n \;\; : \;\; \mathbf{x} \text{ satisfies (7.2b)}\}. \tag{7.3}$$

The following proposition presents the dimension of $P^V(G, K)$.

Proposition 31. *$P^V(G,K)$ is full dimensional.*

Proof. We need to exhibit $n+1$ solutions such that their incidence vectors are affinely independent. Let $S_0 = V$. Clearly, S_0 is a V-UMCFBP solution, since it consists in removing all vertices of the graph. For every vertex $v \in V$, let $S_v = V \setminus \{v\}$. It is also clear that the set $\mathcal{S} = \{S_v : v \in V\}$, which involves keeping in the graph only one vertex, constitutes a set of V-UMCFBP solutions. Indeed, for every commodity $k \in K$, as $s_k \neq t_k$, it is not possible to route any flow from the source to the destination in a graph containing only one vertex. Moreover, the incidence vectors of S_0 and $S_v \in \mathcal{S}$ are affinely independent. Accordingly, S_0 and $\mathcal{S}$ constitute a set of $n+1$ solutions for which their incidence vectors are affinely independent.

□

In what follows, we will be interested in the facial structure of the polytope P^V. More precisely, we focus on the V-target profit inequalities (7.2b) and show that the minimality condition used in Chapter 5 for the MCFBP is not sufficient to ensure that the V-target profit inequalities are facet of the polytope.

Proposition 32. *For every $\boldsymbol{y} \in ext(\mathcal{P}_{umcf})$, inequality $\sum_{v \in \mathcal{V}_S(\boldsymbol{y})} x_v \geq 1$ does not define a facet of $P^V(G,K)$.*

Proof. To prove this proposition, we consider the following graph with 6 vertices and 6 arcs, given by Figure (7.2). All arcs $a \in A$ have a capacity c_a equal to 1 and a cost p_a equal to 0. This graph is associated with three commodities $k_0 = (s_0 = v_0, t_0 = v_3, b_0 = 1, \Gamma_0 = M)$, $k_1 = (s_1 = v_1, t_1 = v_3, b_1 = 1, \Gamma_1 = M)$ and $k_2 = (s_2 = v_1, t_2 = v_5, b_2 = 1, \Gamma_2 = M)$, where M is a constant integer value. The target profit value is set to $M-1$.

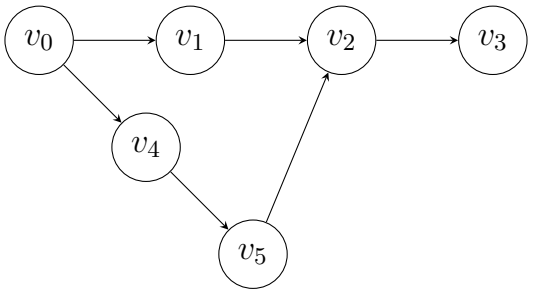


Figure 7.5: Graph with 3 commodities.

The following inequalities are valid for the polytope $P^V(G,K)$.

$$x_{v_0} + x_{v_4} + x_{v_5} + x_{v_2} + x_{v_3} \geq 1 \tag{7.4}$$

$$x_{v_1} + x_{v_2} + x_{v_3} \geq 1 \tag{7.5}$$

$$x_{v_1} + x_{v_0} + x_{v_4} + x_{v_5} \geq 1 \tag{7.6}$$

By summing these inequalities and dividing by 2, we obtain the following inequality:

$$\sum_{v \in V} x_v \geq \frac{3}{2}.$$

As $\sum_{v \in V} x_v$ is an integer for every V-MCFBP solution, the following inequality is valid for the polytope $P^V(G, K)$.

$$\sum_{v \in V} x_v \geq \lceil \frac{3}{2} \rceil \geq 2. \tag{7.7}$$

Accordingly, the V-target profit inequality is dominated by Inequality (7.7), also called *Chvatal-Gomory inequality*, which ends the proof. □

Proposition 32 shows that Inequality $\sum_{v \in \mathcal{V}_S(\boldsymbol{y})} x_v \geq 1$ does not define a facet of $P^V(G, K)$. However, it is worth noticing that when solving the arc-blocker variant in the vertex-augmented graph G^{**} obtained using the graph transformation presented above, the equivalent u-target profit inequality is a facet of $P(G^{**}, K)$, as demonstrated in Chapter 5.

7.3 A Branch-and-Cut algorithm to solve the V-UMCFBP

In this section, we propose a Branch-and-Cut algorithm for the natural Formulation (7.2) with V-target profit inequalities (7.2b) to address the V-UMCFBP. To this end, we first describe the separation problem of the V-target profit inequalities, along with its complexity. We then present various separation strategies aimed at optimizing the algorithm's computational efficiency.

7.3.1 Separation of the V-target profit inequalities

Given a (fractional) solution $\mathbf{x} \in [0, 1]^n$ of the RMP in a B&C node, the separation problem for the V-target profit inequalities (7.2b) requires finding an unsplittable multi-commodity defined by a vector $\boldsymbol{y}^* \in ext(\mathcal{P}_{umcf})$ such that:

$$\sum_{a \in \mathcal{V}_S(\boldsymbol{y}^*)} x_v < 1, \tag{7.8}$$

or to prove that such vector does not exist, i.e., that all V-target profit inequalities (7.2b) are satisfied by the solution $\mathbf{x}$. Thus, it is necessary to find a vector $\boldsymbol{y}^* \in ext(\mathcal{P}_{umcf})$ leading to the minimum value of the left-hand side of (7.8).

Let $\mathbf{z} \in \{0, 1\}^n$ be a vector of binary variables where each variable z_v is equal to 1 if and only if the vertex v is in $\mathcal{V}_S(\boldsymbol{y}^*)$, i.e., if an arc (u, v) is used to route a positive amount of flow. We recall that the total profit of the UMCF associated with $\boldsymbol{y}^*$ is strictly greater than Φ, according to the definition of $\mathcal{P}_{umcf}$. The separation problem can be modeled by the following ILP formulation:

$$\min_{\mathbf{y}\in[0,1]^{d\times m},\mathbf{z}\in\{0,1\}^n} \quad \sum_{v\in V} x_v \cdot z_v \tag{7.9a}$$

$$\sum_{k\in K} y_{k,a}\, b_k \leq c_a, \qquad \forall\, a \in A \tag{7.9b}$$

$$\sum_{a\,\in\,\delta^+(u)} y_{k,a}\, b_k - \sum_{a\,\in\,\delta^-(u)} y_{k,a}\, b_k = \begin{cases} \lambda_k \text{ if } u = s_k, \\ 0 \text{ if } u \in V\backslash\{s_k, t_k\}, \\ -\lambda_k \text{ if } u = t_k \end{cases} \qquad \forall k \in K, \forall u \in V \tag{7.9c}$$

$$y_{k,(u,v)} \leq z_v, \qquad \forall\, k \in K,\ \forall\, (u,v) \in A, \tag{7.9d}$$

$$\sum_{k\in K} \Gamma_k\, \lambda_k - \sum_{a\in A} p_a\, y_{k,a}\, b_k \geq \Phi + 1. \tag{7.9e}$$

The objective function minimizes the left-hand size of (7.8). The constraints are the ones of the UMCFP with the additional constraint "$y_{k,(u,v)} \leq x_v$" that imposes to select an arc $a \in A$ if a positive flow value is routed on this arc for a commodity $k \in K$ and constraint "$\Phi(G) \geq \Phi + 1$" defines the minimum profit of $UMCF$. If the optimal solution value of Model (7.9) is strictly smaller than 1, then a V-target profit inequality maximally violated by $\mathbf{x}$ is found. Otherwise, no V-target profit inequalities are violated by x.

The next proposition characterizes the computational complexity of the separation problem for the V-target profit inequalities (7.2b).

Proposition 33. *The separation problem for the V-target profit inequalities* (7.2b) *is $\mathcal{NP}$-hard for fractional and integer solutions* $\mathbf{x}$ *of the RMP.*

Proof. The decision problem of the separation problem for inequalities (7.2b) asks for finding an unsplittable multi-commodity flow with a total profit greater than or equal to Φ+1. This problem is $\mathcal{NP}$-Complete as stated previously. Accordingly, the optimization version of the problem is $\mathcal{NP}$-hard. □

7.3.2 Branch-and-Cut algorithm

To solve the natural formulation (7.2) with an efficient B&C algorithm, one needs an efficient separation strategy for the V-target profit inequalities (7.2b). To determine a V-target profit inequality violated by an RMP solution $\mathbf{x}$, we propose the separation procedures described in the next sections.

Integer separation of V-target profit inequalities (7.2b)

One way to implement the B&C algorithm is to separate the V-target profit inequalities only for integer LP relaxation points.

This can be done by solving a UMCFP. Indeed, since $x_v \in \{0,1\}, \forall v \in V$, a violated V-target profit inequality can be found if and only if an unsplittable multi-commodity flow defined by a vector $\boldsymbol{y} \in ext(\mathcal{P}_{umcf})$ exists such that the subset $\mathcal{V}_S(\boldsymbol{y}) \subseteq V$ does not contain any blocked vertices. Accordingly, finding a violated V-target profit inequality leads to solve a UMCFP in the vertex non-blocked graph $\mathcal{G}^{\mathcal{V}}_{NB}(\mathbf{x})$. In case the maximum profit $\Psi(\mathcal{G}^{\mathcal{V}}_{NB}(\mathbf{x}))$ is strictly greater than Φ, an unsplittable multicommodity flow defined by a vector $\boldsymbol{y} \in ext(P_{umcf})$ and associated to a violated V-target profit inequality is found. Otherwise, no V-target profit inequalities are violated by $\mathbf{x}$.

An effective separation strategy for fractional points can also significantly enhance the performance of the algorithm. In what follows, we introduce two distinct strategies aimed at separating fractional points.

Exact separation of V-target profit inequalities (7.2b)

Let $(\mathbf{y}^*, \mathbf{x}^*)$ be an optimal solution of Model (7.9). In case the optimal solution value is strictly smaller than 1, then we have detected a violated u-target profit inequality (7.2b) and the following cut is added to the RMP:

$$\sum_{a \in A(\boldsymbol{y}^*)} x_a \geq 1,$$

where $A(\boldsymbol{y}^*)$ is the set of arcs $a \in A$ for which $y^*_{k,a} > 0, \forall k \in K$.

However, solving Model (7.9) could be time-consuming. Therefore, in the following section, we introduce an algorithm to heuristically separate fractional solutions.

Heuristic separation of V-target profit inequalities (7.2b)

We now present a heuristic to solve the exact separation problem of the V-target profit inequalities, i.e., Model (7.9). This heuristic is referred to as `V_U_SP_HEU` since it is based on successive computations of shortest paths.

This heuristic is very similar to the `U_SP_HEU` heuristic introduced in Chapter 6 for solving the arc blocker variant of the UMCFP. However, it has been adapted in this section to address the vertex-blocker variant. Consequently, the approach to identifying a UMCFP with a profit greater than Φ remains very similar, taking into account the blocked vertices rather than the blocked arcs.

As with the `U_SP_HEU`, the `V_U_SP_HEU` operates on the principle of incrementally satisfying a set of commodities to achieve a UMCF of profit greater than Φ, while ensuring that the total blocker cost is less than 1. However, in the case of the `U_SP_HEU`, where a blocker cost is attributed to every arc of the graph, in the `V_U_SP_HEU`, the blocker costs are assigned to vertices of the graph. Therefore, in the `V_U_SP_HEU`, we consider a use-vertex-cost instead of a use-arc-cost. This use-vertex-cost corresponds to the sum of the blocker costs on the vertices in the UMCF. Hence, if a UMCF of profit greater than Φ is found such that the total use-vertex-cost does not exceed 1, then a cut is identified and the corresponding inequality is added to the RMP.

7.4 Computational results

In this section, we present the results of our computational campaign. The aim is to assess the performance of the ILP natural formulations for the vertex variant of the UMCFBP, i.e., the V-UMCFBP.

As in Chapter 6, the experiments are conducted on a processor Intel Core i5-3340M CPU of 2.70GHz × 4. The Branch-and-Cut algorithms are implemented using the `CONCERT TECHNOLOGY` of `CPLEX`. As previously, all computations are performed in a single-thread mode with default values for all `CPLEX` parameters.

We consider the instances presented previously, i.e., the instances from `SNDLIB`, `TELECOM` and `SYNTHETIC` classes. For each instance, we assign to every vertex in the graph a blocker cost. These costs are determined by randomly selecting numbers from respective discrete uniform distributions. Specifically, for all instances, the blocker cost for each vertex is generated from a distribution within the range of $[1, 49]$.

In the remainder of this paper, all computing times are expressed in seconds. We impose a CPU time limit of 600 seconds for every instance. If this time limit is exceeded, it is reported as $t.l.$ in the computational results.

7.4.1 Computational performance of the Branch-and-Cut for the V-UMCFBP

In this section, we study and discuss the computational performance of Model (7.2) for the V-UMCFBP. The aim of this section is to determine the best Branch-and-Cut algorithm to solve the natural formulation for the V-UMCFBP.

The natural formulation is solved using a Branch-and-Cut algorithm where V-target profit inequalities (7.2b) are separated at every node of the branching tree and added to the RMP. It is worth noticing that the RMP is initialized with one constraint that is associated with the maximum-profit unsplittable multicommodity flow in G. This UMCF has, by definition, a profit greater than Φ. We recall that we are interested in separating target profit inequalities both for integer and fractional solutions. As previously discussed, the separation of u-target profit inequalities is $\mathcal{NP}$-Hard for integer and fractional RMP solutions. Therefore, we introduce two distinct separation strategies for integer and fractional points. For integer points, the first strategy entails the separation of integer solutions by solving a UMCFP using `CPLEX` ILP Solver. An alternative approach involves employing a column generation algorithm for solving a UMCFP, proceeding to use the `CPLEX` ILP Solver if no cuts are identified. This latter has demonstrated superior efficiency for the arc-blocker variant. Hence, it is the method we consider for this study. For fractional solutions, the initial method involves the separation of fractional solutions through the exact separation problem (see Model (7.9)). Moreover, we consider a second approach, where a feasible solution to the separation problem (7.9) is considered, rather than the optimal solution, by using the shortest-path based heuristic `V_U_SP_HEU` detailed in Section 7.3.2.

Therefore, we propose two Branch-and-Cut algorithms to solve the natural formulation (7.2). The first algorithm, denoted as $\texttt{INT\_CG\_V} - \texttt{UMCFBP}$, focuses on separat-

ing only integer infeasible points by solving a UMCFP, using the column-generation based approach. The second algorithm, denoted as CG_HEU_V − UMCFBP introduces the separation of fractional infeasible points by applying the shortest-path-based heuristic V_U_SP_HEU.

Tables 7.1 and 7.2 compare performance of these two Branch-and-Cut algorithms on SNDLIB instances, namely, INT_CG_V-UMCFBPand CG_HEU_V-UMCFBP.

					INT_V-UMCFBP				
						time		# lazy	nodes
n	m	d	λ	#	# opt	avg.	max	avg.	avg.
26	84	15	0,2	9	7	t.l.	t.l.	936,21	408,6
			0,6	9	5	198,84	t.l.	757	586
			0,9	9	8	0,83	10.1	44,1	1,76
27	102	16	0,2	6	3	t.l.	t.l.	807	t.l.
			0,6	6	3	t.l.	t.l.	795	t.l.
			0,9	6	6	2,01	7.98	91,78	6,4
39	122	23	0,2	9	2	t.l.	t.l.	600	t.l.
			0,6	9	1	t.l.	t.l.	589,22	t.l.
			0,9	9	9	0,15	0.99	17,95	1.98
	172	23	0,2	3	0	t.l.	t.l.	503	t.l.
			0,6	3	1	t.l.	t.l.	495,55	t.l.
			0,9	3	2	132,09	t.l.	192,14	465
Total				81	47				

Table 7.1: Performance of INT_CG_V-UMCFBPBranch-and-Cut algorithm used to solve Model (7.2b) on SNDLIB instances

The comparative analysis between CG_HEU_V-UMCFBP and INT_CG_V-UMCFBP reveals that CG_HEU_V-UMCFBP consistently outperforms INT_V-UMCFBP, as evidenced by the average computing time and the number of instances solved to proven optimality. This enhancement is primarily attributed to the effective separation of fractional unfeasible points, which subsequently decreases the number of infeasible points to separate. Consequently, this contributes to reducing the number of nodes in the branching tree and hence the overall computing time. To illustrate this statement, let us consider graphs of 26 vertices and 84 arcs. For instance, with INT_V-UMCFBP, the number of integer infeasible points separated (# lazy cuts) averages 936. This number reduces to 57 when incorporating 86 #user cuts. Consequently, the average computing time decreases from exceeding the time limit (600 seconds) to 98 seconds, highlighting the efficiency of CG_HEU_V-UMCFBP.

Furthermore, it is worth noticing that the vertex-blocker variant exhibits better computing times compared to the arc-blocker variant, with 58 instances from SNDLIB solved

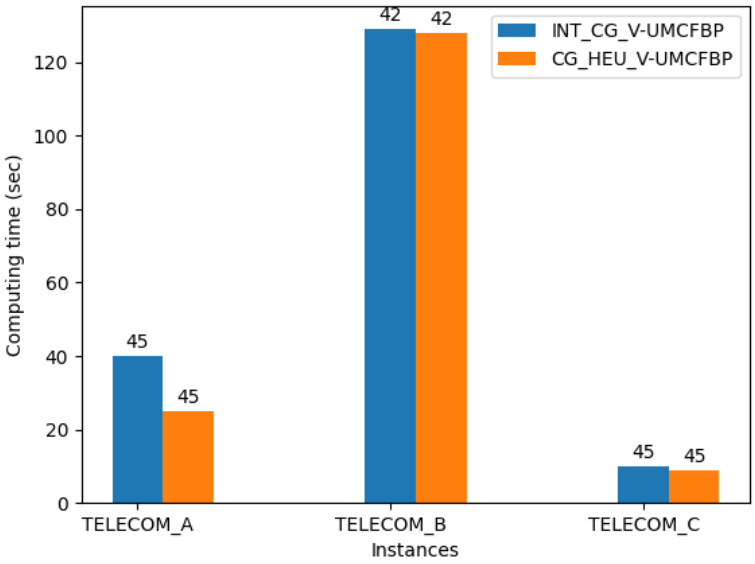
					CG_HEU_V-UMCFBP					
					# opt	time		# lazy	# user	nodes
n	m	d	λ	#	# opt	avg.	max	avg.	avg.	avg.
26	84	15	0,2	9	9	98,82	t.l.	57	86	339,9
			0,6	9	7	0,84	10,2	44	1,75	t.l.
			0,9	9	8	t.l.	t.l.	807	t.l.	807,05
27	102	16	0,2	6	4	2,03	8	91,77	6,5	261,12
			0,6	6	4	t.l.	t.l.	600	t.l.	t.l.
			0,9	6	6	132,1	t.l.	192,15	465	133,1
39	122	23	0,2	9	4	198,85	t.l.	757	587	339,87
			0,6	9	3	0,85	10,1	44,2	1,77	t.l.
			0,9	9	9	t.l.	t.l.	807	t.l.	808
	172	23	0,2	3	1	2,02	7,97	91,79	6,3	262
			0,6	3	1	198,81	t.l.	758	588	134,2
			0,9	3	2	0,83	10,15	44,1	1,76	340
Total				81	58					

Table 7.2: Performance of CG_HEU_V-UMCFBP Branch-and-Cut algorithm used to solve Model (7.2b) on SNDLIB instances

to proven optimality, against 37 for the arc-blocker problem. This observation is consistent, as removing vertices instead of arcs results in reduced combinatorial complexity. This reduction is directly associated with the number of vertices in the graph, which consistently tends to be smaller than the number of arcs.

In Figure 7.4.1, we present a comparative analysis of the two algorithms for solving the V-UMCFBP, i.e., INT_CG_V-UMCFBP and CG_HEU_V-UMCFBP on TELECOM instances. Specifically, Figure 7.4.1 illustrates a bar chart depicting the average computing time of each algorithm on TELECOM_A, TELECOM_B and TELECOM_C instances. Each category of instances is represented by two adjacent bars, corresponding to an algorithm. Additionnaly, above each bar, we report the number of instances solved to proven optimality. We recall that each group of instances counts a total of 45 instances. This analysis reveals that for the vertex variant, the shortest-path based heuristic proposed demonstrates limited efficiency according to the TELECOM instances. Although the average computing time may not be substantial and only 3 instances remain unsolved, this observation underscores the potential for improvement through the refinement of current methodologies or the exploration of alternative heuristics for the vertex-blocker variant.

Figure 7.6: Performance comparison of the natural formulation (7.2) for the V-UMCFBP on `TELECOM` instances



7.5 Concluding remarks and perspectives

In this chapter, our attention shifts to another variant of the network flow blocker problem, known as the network flow vertex-blocker problem. Unlike its counterpart, this variant targets the removal of vertices in graphs rather than arcs. Specifically, we investigate the unsplittable multicommodity flow problem within this context. Thus, we introduce a natural formulation for the unsplittable multicommodity flow vertex-blocker problem (V-UMCFBP), characterized by an exponential family of constraints, namely the V-target profit inequalities. To address this problem, we develop a tailored Branch-and-Cut algorithm featuring several separation procedures, adapted from the arc-blocker variant.

Furthermore, after describing the polyhedra of solutions for the proposed model, we illustrate that, unlike the arc-blocker variant, a V-target profit inequality does not define a facet under the conditions of the arc-blocker variant. However, as a direction for future research, further exploration could be pursued to identify a new valid inequality. A preliminary investigation in this direction is outlined below.

New valid inequality Let us go back to the Chvatal-Gomory inequality (7.7) and propose a generalization.

Let us consider the example graph represented in Figure 7.2. Suppose that there exists a vertex $u \in V$ such that for a commodity $k \in K$, the set of arcs $\{(s_k, u), (u, t_k)\}$ constitutes a UMCF with a profit greater than 2, then the following inequality is valid for Formulation (7.2):

$$\sum_{v \in \mathcal{V}_S(\boldsymbol{y})} x_v + x_u \geq 2, \forall \boldsymbol{y} \in \mathcal{P}_{umcf} \tag{7.10}$$

More generally, if there exists a subset of commodities $K' \subseteq K$, such that for each commodity $k \in K'$, the source s_k and the destination t_k are connected by a vertex u_k, then the subsequent family of inequalities holds true for Formulation (7.2):

$$\sum_{v \in \mathcal{V}_S(\boldsymbol{y})} x_v + \sum_{k \in K'} x_{u_k} \geq 1 + q, \forall \boldsymbol{y} \in \mathcal{P}_{umcf}, \tag{7.11}$$

where q is the number of vertices $u_k, k \in K'$.

Conclusion

In this dissertation, we are interested in network flow blocker problems. More precisely, we focus on blocker problems applied to the multicommodity flow problem and its variants.

At the beginning of the thesis, we present practical applications for studying network flow blocker problems, with a primary focus on telecommunication networks. Specifically, one direct application aims to enhance resilience against anomalies to ensure the robustness and reliability of telecommunication networks. Therefore, we design in this thesis algorithms tailored for this purpose. Furthermore, we extend our scope to encompass another application known as network-wide monitoring. Within this context, we study the sketch assignment problem and propose solution methods.

In the first part of the thesis, we delve into the maximum flow blocker problem (MFBP). The motivation behind exploring this problem is diverse. First, it serves as a specific instance of the multicommodity flow blocker problem. Second, its relevance stems from the specific structure of the subproblem which is a maximum flow problem. Third, the extensive existing literature surrounding this problem underscores its significance and potential for further investigation. To tackle this problem, we propose several integer linear programming formulations. Some have an exponential family of constraints and they are accordingly solved using a tailored Branch-and-Cut algorithm. Another formulation having a polynomial number of variables and constraints has been designed. These formulations are compared theoretically by studying the strength of the LP relaxations and through a comprehensive computational study. Moreover, we demonstrate a structural link between the solutions of the blocker and the interdiction variant of the maximum flow problem. This is the first time a relationship between a blocker and an interdiction problem applied to another optimization problem has been clarified. This theoretical result enables us to derive another algorithm for addressing the MFBP.

In the second part of the thesis, we delve into the multicommodity flow blocker problem, distinguishing two scenarios: one where the flow is splittable (MCFBP) and the other where the flow is splittable, meaning that it is routed through a unique path for each commodity (UMCFBP). To tackle this challenge, we introduce a concise and well-understood formulation within the natural space of the blocker variables. This formulation, featuring an exponential number of constraints, can be adapted for several versions of the multicommodity flow blocker problem. For each variant (splittable and unsplittable), we study the associated separation problem and propose several approaches. We then present a tailored Branch-and-Cut algorithm to solve the MCFBP and the UMCFBP. Moreover, to enhance the robustness of the model, we present a comprehensive polyhedral analysis. The various approaches and potential enhancements of

the Branch-and-Cut algorithms are compared thanks to an extensive computational analysis on sets of synthetic and real-world instances. The primary objective of this study is to evaluate, for each variant of the problem, the performance of the proposed formulation and to identify the characteristics of instances that can be solved to proven optimality.

Finally, in the third part of the thesis, we extend our work to address the vertex blocker variant of the unsplittable multicommodity flow problem (V-UMCFBP). This variant entails the removal of vertices in the graph rather than arcs. We demonstrate the existence of a relationship between the two variants, using graph transformations. Specifically, we establish that solving one variant is equivalent to solving another in a transformed graph. Additionally, we introduce a dedicated model for the V-UMCFBP featuring an exponential number of constraints, solved using a customized Branch-and-Cut algorithm. Moreover, we investigate the properties of the associated polyhedra.

There are many directions in which the research in this dissertation can be continued. For the maximum flow blocker problem, delving into the polyhedra of the proposed formulations allows for the identification of conditions under which the inequalities define facets and the potential discovery of new valid inequalities. Moreover, exploring deeper the relationship between blocker and interdiction could yield valuable insights, potentially extending its applicability to other problems.

For the multicommodity flow blocker problem, future works could focus on two key directions. The first one concerns the algorithmic aspect that involves enhancing existing heuristics or implementing more advanced branching strategies. The second direction is more theoretical with a deeper exploration of the polyhedra associated with the formulations proposed for the different variants of the problem. An initial line of investigation could be to further develop the work done on the vertex-blocker variant, by initially evaluating the effectiveness of the identified valid inequality, and subsequently adapting it for the arc-blocker variant.

Finally, natural directions for future research include integrating real-world constraints from telecommunication networks. Specifically, this may entail generalizing anomalies to consider traffic peaks, for example, or incorporating additional constraints related to network monitoring. Moreover, proposing optimal routing strategies that account for anomalies is another option to investigate. Achieving this could involve redefining the actions of the blocker.

Bibliography

Anup Agarwal, Zaoxing Liu, and Srinivasan Seshan. HeteroSketch: Coordinating network-wide monitoring in heterogeneous and dynamic networks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 719–741, Renton, WA, April 2022. USENIX Association. ISBN 978-1-939133-27-4.

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice hall, N.Y, 1993.

F. Alvelos and J. Carvalho. Comparing branch-and-price algorithms for the unsplittable multicommodity flow problem. 01 2003.

A. A. Assad. Multicommodity network flows—a survey. *Networks*, 8(1):37–91, 1978.

Nikitas Assimakopoulos. A network interdiction model for hospital infection control. *Computers in Biology and Medicine*, 17(6):413–422, 1987. ISSN 0010-4825. doi: https://doi.org/10.1016/0010-4825(87)90060-6. URL `https://www.sciencedirect.com/science/article/pii/0010482587900606`.

Sabina Barakovic, Jasmina Barakovic Husic, and Enida Cero Dinarević. Iot's tiny steps towards 5g: Telco's perspective. *Symmetry*, 9, 10 2017. doi: 10.3390/sym9100213.

Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. Using Branch-and-Price-and-Cut to Solve Origin-Destination Integer Multicommodity Flow Problems. *Operations Research*, 48(2):318–326, 2000.

Y. Beck, I. Ljubic, and M. Schmidt. A survey on bilevel optimization under uncertainty. *European Journal of Operational Research*, 311(2), 2023.

A. Benhamiche, M. Chopin, and S. Martin. Unsplittable shortest path routing: Extended model and matheuristic. *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2023.

I. Bentoumi, F.Furini, A. R. Mahjoub, and S. Martin. A branch-and-cut algorithm to solve the multi-commodity flow blocker problem. *22éme congrés de la société Francaise de Recherche Opérationnelle et d'Aide à la Décision*, 2021.

I. Bentoumi, F.Furini, A. R. Mahjoub, and S. Martin. A branch-and-benders-cut approach to solve the maximum flow blocker problem. *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 674–677, 2023a.

I. Bentoumi, F.Furini, A. R. Mahjoub, and S. Martin. Integer linear formulations for the maximum flow blocker problem. *Book of abstracts PGMO DAYS 2023*, 2023b.

I. Bentoumi, F.Furini, A. R. Mahjoub, and S. Martin. Optimization methods for the multi-commodity flow blocker problem. *24éme congrés de la société Francaise de Recherche Opérationnelle et d'Aide à la Décision*, 2023c.

Valerio Bruschi, Ran Ben Basat, Zaoxing Liu, Gianni Antichi, Giuseppe Bianchi, and Michael Mitzenmacher. Discovering the heavy hitters with disaggregated sketches. In *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '20, page 536–537, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379489. doi: 10.1145/3386367.3431674. URL `https://doi.org/10.1145/3386367.3431674`.

C. Burch, R. Carr, S. O. Krumke, M. Marathe, C. Phillips, and E. Sundberg. A decomposition-based pseudoapproximation algorithm for network flow inhibition. *Operations Research/ Computer Science Interfaces Series*, **22**:51–68, 2006. https://doi.org/10.1007/0-306-48109-X_3.

Paola Cappanera and Maria Scaparra. Optimal allocation of protective resources in shortest-path networks. *Transportation Science*, 45:64–80, 02 2011. doi: 10.2307/23017638.

G. Castellano, M. Gallo, S. Martin, and I. Bentoumi. Apparatus and methods for network-wide sketching, 2023.

Stephen A. Cook. The complexity of theorem-proving procedures. *Proceedings of the third annual ACM symposium on Theory of computing*, 1971.

Jr. H. W. Corley and H. Chang. Finding the n most vital nodes in a flow network. *Management Science*, **21**(3):362–364, 1974. https://doi.org/10.1287/mnsc.21.3.362.

K. J. Cormican, D. P. Morton, and R. K. Wood. Stochastic network interdiction. *Operations Research*, **46**(2):184–197, 1998. https://doi.org/10.1287/opre.46.2.184.

Graham Cormode. Count-min sketch. 01 2009. doi: 10.1007/978-0-387-39940-9_87.

Graham Cormode. Summary data structures for massive data. In Paola Bonizzoni, Vasco Brattka, and Benedikt Löwe, editors, *The Nature of Computation. Logic, Algorithms, Applications*, pages 78–86, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39053-1.

Graham Cormode. Data sketching. *Communications of the ACM*, 60:48–55, 08 2017. doi: 10.1145/3080008.

E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiway cuts (extended abstract). STOC '92, page 241–251. Association for Computing Machinery, 1992.

George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

S. Dempe. *Bilevel Optimization: Theory, Algorithms, Applications and a Bibliography*, chapter 20, pages 581–672. Springer International Publishing, 2020. https://doi.org/10.1007/978-3-030-52119-6_20.

Martin Dietzfelbinger. *Universal Hashing via Integer Arithmetic Without Primes, Revisited*, pages 257–279. Springer International Publishing, Cham, 2018. ISBN 978-3-319-98355-4. doi: 10.1007/978-3-319-98355-4_15. URL `https://doi.org/10.1007/978-3-319-98355-4_15`.

Y. Dinitz, N. Garg, and M.X. Goemans. On the single-source unsplittable flow problem. In *Proceedings 39th Annual Symposium on Foundations of Computer Science*, pages 290–299, 1998.

J. Edmonds. Covers and packings in a family of sets. *Bulletin of the American Mathematical Society,*, 68(5):494–499, 1971.

J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, **19**(2):248–264, 2003. https://doi.org/10.1007/3-540-36478-1_4.

Jack Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, page 125, 1965.

S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 184–193, 1975. doi: 10.1109/SFCS.1975.21.

Mohammad Fathi and H. Bevrani. *Convex Programming*, pages 69–93. 01 2019. ISBN 978-3-030-05308-6. doi: 10.1007/978-3-030-05309-3_4.

M. Fischetti and I. Ljubic. A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research*, **65**(6):1615–1637, 2017.

M. Fischetti, I. Ljubic, and S. Monaci. On the use of intersection cuts for bilevel optimization. *Math. Program.*, 2018.

Jr. L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, **8**:399–404, 1956. https://doi.org/10.4153/CJM-1956-045-5.

L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

F. Furini, I. Ljubić, S. Martin, and P. San Segundo. The maximum clique interdiction problem. *European Journal of Operational Research*, **277**(1):112–127, 2019. https://10.1016/j.ejor.2019.02.028.

L.F. Carvalho G. Fernandes, J.J.P.C. Rodrigues. A comprehensive survey on network anomaly detection. *Telecommun Syst*, 70:447–489, 2019.

Carlos García-Martínez, Christian Blum, Francisco Rodríguez, and Manuel Lozano. The firefighter problem: Empirical results on random graphs. *Computers & Operations Research*, 60:55–66, 08 2015. doi: 10.1016/j.cor.2015.02.004.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman, San Francisco, 1979.

I. Ghafir, V. Prenosil, and M. Hammoudeh. A survey on network security monitoring systems. pages 77–82, 2016.

A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, **35**(4):921–940, 1988. https://doi.org/10.1145/48014.61051.

R. Grappe, M. Lacroix, and S. Martin. The multiple pairs shortest path problem for sparse graphs: Exact algorithms. *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2023.

Martin Grötschel, Lovász László, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 06 1981. doi: 10.1007/BF02579273.

David Gutierrez-Estevez, Marco Gramaglia, Antonio Domenico, Ghina Dandachi, Sina Khatibi, Dimitris Tsolkas, Irina Balan, Andres Garcia-Saavedra, Uri Elzur, and Yue Wang. Artificial intelligence for elastic management and orchestration of 5g networks. *IEEE Wireless Communications*, PP:1–8, 08 2019. doi: 10.1109/MWC.2019.1800498.

Michael Held and Richard M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162, 1970.

T. C. Hu. Multi-commodity network flows. *Operations Research*, 11(3):344–360, 1963.

N. Huin, J. Leguay, S. Martin, and P. Medagliani. Routing and slot allocation in 5g hard slicing. *Computer Communications*, 201:72–90, 2023.

E. Israeli and K. R. Wood. Shortest-path network interdiction. *Networks*, **40**(2):97–111, 2002a. https://doi.org/10.1002/net.10039.

E. Israeli and K. R. Wood. Shortest-path network interdiction. *Networks*, **40**(2):97–111, 2002b.

G. Junior, J. Rodrigues, L. Carvalho, J. Al-Muhtadi, and M. Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, **70**(3):447–489, 2019. https://doi.org/10.1007/s11235-018-0475-8.

A. Juttner, B. Szviatovski, I. Mecs, and Z. Rajko. Lagrange relaxation based method for the qos routing problem. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society*, volume 2, pages 859–868, 2001.

Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972. ISBN 978-1-4684-2001-2. doi: 10.1007/978-1-4684-2001-2_9. URL `https://doi.org/10.1007/978-1-4684-2001-2_9`.

J.L. Kennington. A survey of linear cost multicommodity network flows. *Operations Research,*, 26(2):209–236, 1978.

L. Khachiyan, K. Borys E. Boros, K. Elbassioni, V. Gurvich, G. Rudolf, and J. Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, **43**:204–233, 2008.

Jon M. Kleinberg. Approximation algorithms for disjoint paths problems. Ph.D. thesis, MIT, Cambridge MA, 1996.

Thomas Kleinert, Martine Labbé, Ivana Ljubić, and Martin Schmidt. A survey on mixed-integer programming techniques in bilevel optimization. *EURO Journal on Computational Optimization*, 9:100007, 2021. ISSN 2192-4406.

Stavros G. Kolliopoulos and Clifford Stein. Approximation algorithms for single-source unsplittable flow. *SIAM Journal on Computing*, 31(3):919–946, 2001.

J. Krolikowski, S. Martin, P. Medagliani, J. Leguay, S. Chen, X. Chang, and X. Geng. Joint routing and scheduling for large-scale deterministic ip networks. *Computer Communications*, 165:33–42, 2021a.

Jonatan Krolikowski, Sebastien Martin, Paolo Medagliani, Jeremie Leguay, Shuang Chen, Xiaodong Chang, and Xuesong Geng. Joint routing and scheduling for large-scale deterministic ip networks. *Computer Communications*, 165:33–42, 01 2021b. doi: 10.1016/j.comcom.2020.10.016.

S. O. Krumke, H. Noltemeier, S. Schwarz, HC. Wirth, and R. Ravi. Flow improvement and network flows with fixed costs. pages 158–167, 1999.

Martine Labbé, Patrice Marcotte, and Gilles Savard. A bilevel model of taxation and its application to optimal highway pricing. *Management Science*, 44:1608–1622, 12 1998. doi: 10.1287/mnsc.44.12.1608.

Ashwin Lall, Vyas Sekar, Mitsunori Ogihara, Jun Xu, and Hui Zhang. Data streaming algorithms for estimating entropy of network traffic. volume 34, pages 145–156, 06 2006. doi: 10.1145/1140277.1140295.

Mohammed Lalou, Mohammed Amin Tahraoui, and Hamamache Kheddouci. The critical node detection problem in networks: A survey. *Computer Science Review*, 28:92–117, 2018. ISSN 1574-0137. doi: https://doi.org/10.1016/j.cosrev.2018.02.002. URL `https://www.sciencedirect.com/science/article/pii/S1574013716302416`.

Pierre Laroche, Franc Marchetti, Sébastien Martin, Anass Nagih, and Zsuzsanna Róka. Multiple bipartite complete matching vertex blocker problem: Complexity, polyhedral analysis and branch-and-cut. *Discrete Optimization*, 35:100551, 2020. ISSN 1572-5286. doi: https://doi.org/10.1016/j.disopt.2019.100551. URL `https://www.sciencedirect.com/science/article/pii/S1572528617301469`.

T. Li, J. Ma, Q. Pei, Y. Shen, and C. Sun. Anomalies detection of routers based on multiple information learning. In *2018 International Conference on Networking and Network Applications (NaNA)*, pages 206–211. IEEE Computer Society, 2018.

C. Lim and J. Cole Smith. Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions*, 39(1):15–26, 2007a.

Churlzu Lim and J. Smith. Algorithms for discrete and continuous multicommodity flow network interdiction problems. *Iie Transactions*, 39:15–26, 01 2007b. doi: 10.1080/07408170600729192.

Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 101–114, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341936. doi: 10.1145/2934872.2934906. URL `https://doi.org/10.1145/2934872.2934906`.

Y. Magnouche and S. Martin. Most vital vertices for the shortest $s-t$ path problem: complexity and branch-and-cut algorithm. *Optimization Letters*, **14**:2039–2053, 2020. doi: https://10.1007/s11590-019-01527-5.

P. Marcotte and D. L. Zhu. Exact and inexact penalty methods for the generalized bilevel programming problem. *Math. Program.*, 74(2):141–157, aug 1996. ISSN 0025-5610.

M. P. Martin and S. Martin. Unsplittable multi-commodity flow problem via quantum computing. *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2023.

S. Martin, P. Medagliani, and J. Leguay. Network slicing for deterministic latency. *2021 17th International Conference on Network and Service Management (CNSM)*, pages 572–577, 2021.

S. Martin, Y. Magnouche, C C. Juvigny, and J. Leguay. Constrained shortest path tour problem: Branch-and-price algorithm. *Computers & Operations Research*, 144, 2022.

N. Olver and L. Vegh. A simpler and faster strongly polynomial algorithm for generalized flow maximization. *Journal of the ACM*, 67, 2016.

S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. In *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, April 2007. URL `http://www.zib.de/orlowski/Paper/OrlowskiPioroTomaszewskiWessaely2007-SNDlib-INOC.pdf.gz`. http://sndlib.zib.de, extended version accepted in Networks, 2009.

A. Ouorou, P. Mahey, and J.-Ph. Vial. A survey of algorithms for convex multicommodity flow problems. *Management Science*, 46(1):126–147, 2000.

Antonio Pescapé, Dario Rossi, Davide Tammaro, and Silvio Valenti. On the impact of sampling on traffic monitoring and analysis. In *2010 22nd International Teletraffic Congress (lTC 22)*, pages 1–8, 2010. doi: 10.1109/ITC.2010.5608718.

Marco Polverini, Antonio Cianfrani, Marco Listanti, and Andrea Baiocchi. Routing perturbation for traffic matrix evaluation in a segment routing network. *IEEE Transactions on Network and Service Management*, 15(4):1645–1660, 2018. doi: 10.1109/TNSM.2018.2862423.

H. D. Ratliff, G. T. Sicilia, and S. H. Lubore. Finding the n most vital links in a flow network. *Management Science*, **21**(5):531–539, 1975. https://doi.org/10.1287/mnsc.21.5.531.

J. Royset and R. K. Wood. Solving the bi-objective maximum-flow network-interdiction problem. *INFORMS Journal on Computing*, **19**(2):175–184, 2007. https://doi.org/10.1287/ijoc.1060.0191.

A. Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, **91**(3):437–445, 2002. https://doi.org/10.1007/s101070100259.

A. Schrijver. *Combinatorial Optimization Polyhedra and Efficiency.* Springer-Verlag Berlin Heidelberg, 2003.

Lawrence Snyder, Zumbul Atan, Peng Peng, Ying Rong, Amanda Schmitt, and Burcu Sinsoysal. Or/ms models for supply chain disruptions: A review. *IIE Transactions*, 48:89–109, 11 2016. doi: 10.2139/ssrn.1689882.

H. Stackelberg. *The Theory of the Market Economy.* William Hodge, 1952.

E. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.

Eva Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, **5**(3):247–255, 1985.

I. L. Wang. Multicommodity network flows: A survey, part i: Applications and formulations. 15:145–153, 12 2018a.

I. L. Wang. Multicommodity network flows: A survey, part ii: Solution methods. 15: 155–173, 12 2018b.

N. Wei and J. Walteros. Integer programming methods for solving binary interdiction games. *European Journal of Operations reseach*, 9:456–469, 2022.

R. K. Wood. Deterministic network interdiction. *Mathematical and Computer Modeling*, **17**(2):1–18, 1993. https://doi.org/10.1016/0895-7177(93)90236-R.

R. K. Wood. Bilevel network interdiction models: Formulations and solutions. *Wiley Encyclopedia of Operations Research and Management Science*, **174**:1–11, 2011. https://doi.org/10.1002/9780470400531.eorms0932.

P. Xu. Three essays on bilevel optimization algorithms and applications. *PhD thesis, Iowa State University*, 2012.

R. Zenklusen. Matching interdiction. *Discrete Applied Mathematics*, **158**(15):1676–1690, 2010. https://doi.org/10.1016/j.dam.2010.06.006.

J. Zhang, Y. Magnouche, S. Martin, A. Fressancourt, and J. C. Beck. The multi-commodity flow problem with disjoint signaling paths: A branch-and-benders-cut algorithm. *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2023.

J. Zhang, Y. Magnouche, P. Bauguion, S. Martin, and J. C. Beck. Computing bipath multicommodity flows with constraint programming–based branch-and-price-and-cut. *INFORMS Journal on Computing*, 2024.

Ying Zhang. An adaptive flow counting method for anomaly detection in sdn. pages 25–30, 12 2013. doi: 10.1145/2535372.2535411.

RÉSUMÉ

Les réseaux de télécommunication sont des systèmes complexes dans lesquels s'inscrivent des problèmes d'optimisation combinatoire difficiles. Face à une demande croissante, l'efficacité de ces réseaux devient cruciale, notamment en termes de délai et de résilience face aux anomalies. Cette thèse se concentre sur l'analyse de la résilience des réseaux, avec pour objectif principal de déterminer le nombre maximal d'anomalies que le réseau peut supporter tout en maintenant sa fonctionnalité selon des critères spécifiques.
Ce défi relève d'une classe de problèmes d'optimisation connus sous le nom de problèmes de bloqueur sur les flots. Notre travail se focalise particulièrement sur les problèmes de multiflots rencontrés dans les réseaux de télécommunication, caractérisés par des contraintes complexes de satisfaction des demandes. Nous nous intéressons aux demandes ayant différentes formes de trafic et prédites avec des méthodes d'apprentissage automatique. Le problème étudié est appelé problème de bloqueur sur les multiflots. Pour répondre à cette problématique, nous utilisons des outils d'optimisation combinatoire, explorant diverses approches, y compris les techniques bi-niveau, l'approche polyèdrale et les algorithmes de branchement.
Nous abordons initialement le problème du bloqueur sur le flot maximal. Pour le résoudre, nous proposons plusieurs formulations de programmation linéaire entière, ainsi qu'une technique dérivée d'un résultat théorique établissant un lien structurel avec un problème existant dans la littérature. Par la suite, nous étendons notre travail pour traiter la notion de bloqueur sur les multiflots, en couvrant plusieurs variantes. Pour ce faire, nous introduisons une formulation avec une famille exponentielle de contraintes, résolue à l'aide d'un algorithme de branchement et de coupes. De plus, nous développons une approche polyèdrale pour renforcer la robustesse du modèle. Les performances des méthodes exactes proposées pour résoudre le deux problèmes décrits sont évaluées à travers une étude expérimentale approfondie.

MOTS CLÉS

Optimisation combinatoire, flot maximum, multiflots, bloqueur, interdiction, optimisation biniveau

ABSTRACT

Telecommunication networks are complex systems in which hard combinatorial optimization problems must be solved. With the increasing demand, telecommunication networks have to be efficient, especially in terms of time lag and failure tolerance. In this context, this thesis will focus on the resilience analysis of a network. More precisely, the primary goal of this study is to determine the maximum number of anomalies that may occur in the network while guaranteeing its functionality depending on a crucial property that needs to be maintained.
This challenge belongs to a class of optimization problems called network flow blocker problems. The initial purpose of this work is to focus on multi-commodity flow problems arising in telecommunication networks with complex demand satisfaction constraints. We are interested in demands, also called commodities, having different shapes of traffic and predicted with machine learning methods. The associated network flow blocker problem is called multicommodity flow blocker problem (MCFBP). To tackle this challenge, we use combinatorial optimization tools, delving into various approaches including bilevel techniques that exploit the bilevel nature of network flow blocker problems, a polyhedral approach and branching algorithms.
We first focus on the maximum flow blocker problem (MFBP), which is a particular case of the MCFBP. To solve the MFBP, we propose several Integer Linear Programming (ILP) formulations and a technique derived from a theoretical result that establishes a structural link between the MFBP and another problem existing in the literature. We then expand our work to address the blocker notion of the multicommodity flow problem, covering several variants. For this problem, we introduce an ILP formulation featuring an exponential number of constraints and solved using a tailored branch-and-cut algorithm. In addition, we enhance the model's robustness by studying its polyhedra. Performance of the exact methods proposed to solve the MFBP and the MCFBP are evaluated through an extensive computational campaign involving a set of synthetic and real-world instances.

KEYWORDS

Combinatorial optimization, maximum flow, multicommodity flow, blocker, interdiction, bilevel optimization