# Fixed charge network design problem with user-optimal flows

Ikram Bouras

▶ **To cite this version:**

Ikram Bouras. Fixed charge network design problem with user-optimal flows. Other [cs.OH]. Université Montpellier, 2019. English. NNT : 2019MONTS136 . tel-02929326

**HAL Id: tel-02929326**
**https://theses.hal.science/tel-02929326**

Submitted on 3 Sep 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

**THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITE DE MONTPELLIER**

**En Informatique**

**École doctorale I2S Information, Structures, Systèmes**

**Unité de recherche LIRMM, UMR 5506**

# Fixed Charge Network Design Problem with User-Optimal flows


**Présentée par Ikram Bouras**

**Le 20/12/2019**

**Sous la direction de Michael Poss et Rosa Figueiredo**


**Devant le jury composé de**

| | |
|---|---|
| Dritan Nace, Prof, Heudiasyc, UTC Compiègne | Rapporteur |
| Luce Brotcorne, Dr, INRIA, équipe-projet INOCS, Lille | Rapporteure |
| Safia Kedad-Sidhoum, Prof, CNAM-CEDRIC, Bordeaux | Examinatrice (Présidente du jury) |
| Boris Detienne, MCF, IMB, Bordeaux | Examinateur |
| Michael Poss, CR CNRS, LIRMM, Montpellier | Directeur de thèse |
| Rosa Figueiredo, MCF, LIA, Avignon | Co-encadrante |

UNIVERSITÉ DE MONTPELLIER

# Abstract

This thesis addresses a class of bi-level network design problems. We are interested in investigating applications from different domains and in developing exact algorithms to solve the corresponding bi-level network problem. In particular, we study a bi-level network design problem where the leader selects a part of the network to be activated, then, in the second level, the solution must be optimal for a network flow problem in the selected sub-network. In this thesis, three applications of this problem are studied: hazmats transportation, telecommunication, and social networks analysis. The second level problem in the first and the last applications is a shortest path problem while a minimum cost flow is required in the second application.

The first studied problem is the fixed charge network design problem with shortest path constraints, which is modeled as a bi-level program and can be applied in hazardous transportation. For this problem, we propose two new binary integer programming (BILP) formulations inspired by path and cycle inequalities. We incorporate these formulations in a branch-and-cut algorithm and another cutting-plane based method. Numerical experiments are performed on real instances, and random data sets generated with different criteria to examine the difficulty of the instances. The results show that the proposed cutting plane algorithms can solve up to 19% more instances than the compact formulations.

The second studied problem is the energy-aware traffic engineering while using multi-path routing to minimize link capacity utilization in ISP backbone networks. We propose a bi-level optimization model where the upper level represents the energy management function, and the lower one refers to the deployed multi-path routing protocol. Then, we reformulate it as a one-level MILP replacing the second level problem by different sets of flow optimality conditions. We further use these formulations to solve the problem with classical cutting plane and branch-and-cut algorithms. The computational experiments are performed on real instances to compare the proposed algorithms and to evaluate the efficiency of our model against existing single-path and multi-objective models.

Finally, we study the problem of maximization influence in signed social networks. To the best of our knowledge, it is the first time that this problem is modeled as a bi-level programming problem. We reformulate the problem as one-level MILP models using three different optimality conditions of the shortest path problem appearing in the second level. These formulations are strengthened by adding a set of valid inequalities. Computational experiments are performed using random instances to compare the different proposed formulations. Finally, explicit solutions and bounds are proposed for particular cases of instances.

***Keywords***— Network design, Bi-level programming, Integer programming, Transportation networks, Telecommunication networks, Social networks

# Résumé

Cette thèse s'adresse à la classe des problèmes de conception de réseaux bi-niveaux. Nous nous sommes intéressés à l'étude des applications dans différents domaines et au développement d'algorithmes exacts pour la résolution des problème de réseaux bi-niveau correspondants. En particulier, nous avons étudié le problème de conception de réseau bi-niveau dans lequel le "leader" sélectionne une partie du réseau à activer, puis, dans le deuxième niveau, la solution doit être optimale pour un problème de flot dans le sous-réseau sélectionné. Dans cette thèse, trois applications de ce problème sont étudiées : le transport de matières dangereuses, les réseaux de télécommunication et les réseaux sociaux. Les problèmes de deuxième niveau de la première et la dernière application sont des problèmes de plus court chemin alors qu'un flot de coûts minimum est requis dans la deuxième application.

Le premier problème étudié est le problème de conception de réseau avec coût fixe avec contraintes de plus court chemin. Le problème est modélisé comme un programme bi-niveaux qui peut être appliqué dans le transport des matières dangereuses. Pour ce problème, nous proposons deux nouvelles formulations de programmation en nombres entiers (PLNE) inspirées par des inégalités de chemin et de cycle. Nous incorporons ces formulations dans des algorithmes de branch-and-cut et de plans coupants. Des tests numériques sont effectués sur des instances réelles et sur un ensembles d'instances aléatoires

qui sont générées avec différents critères pour examiner la difficulté de ces instances. Les résultats montrent que les algorithmes de plan coupants proposés peuvent résoudre jusqu'à 19 % d'instances de plus que les formulations compactes.

La deuxième application concerne la gestion de la consommation d'énergie dans les réseaux de télécommunication en utilisant un protocole de routage multi-chemins pour minimiser la capacité des liens utilisée. Nous proposons un modèle d'optimisation bi-niveaux dans lequel le premier niveau représente la fonction de gestion de l'énergie et le deuxième niveau est un protocole de routage multi-chemins. Ensuite, le problème est reformulé par des formulations PLNE en remplaçant le problème du deuxième niveau par ses conditions d'optimalité. Ces formulations sont utilisées pour résoudre le problème avec les algorithmes classiques de plans coupants et de branch-and-cut. Les expérimentations sont effectuées sur des instances réelles afin de comparer les algorithmes proposés et d'évaluer l'efficacité de notre modèle par rapport à des modèles alternatifs proposés dans la littérature.

Enfin, nous étudions le problème de la maximisation d'influence dans les réseaux sociaux signés. A notre connaissance, c'est la première fois que ce problème est considéré comme un problème de programmation à deux niveaux. Nous reformulons le problème en modèles PLNE à un niveau en utilisant trois différentes conditions d'optimalité du problème de plus court chemin apparaissant dans le deuxième niveau. Ces formulations sont renforcées en ajoutant un ensemble d'inégalités valides. Des tests numériques sont effectués sur des instances aléatoires pour comparer les différentes formulations proposées. Enfin, des solutions optimales en temps polynomial sont proposées

pour des cas particuliers des graphes.

***Mots clés***— Conception de réseaux, Programmation bi-niveau, PLNE, Réseaux de transport, Réseaux de télécommunication, Réseaux sociaux.

# Acknowledgements

This thesis becomes a reality with the kind support and help of many individuals. I would like to extend my thanks to all of them.

First and foremost, I would like to express my sincere gratitude to my advisors **Michael Poss** and **Rosa figueiredo** for the continuous support of my Ph.D. study and research, for their patience, motivation, immense knowledge, and their advice throughout these three years of research. They brought me a deeper understanding of the different aspects of the subject and the research. I could not have imagined having better advisors and mentors for my Ph.D study.

I would like to thank **Fen Zhou** for the information he gave me about telecommunication networks and for all the help during my two first years. I am also very grateful to **Cristina Requejo** for her collaboration on social network problems and for the support during my stay in Aveiro, Portugal.

Besides my supervisors, I would like to thank **Prof. Safia Kedad-Sidhoum** for her kind acceptance to be the president of my Ph.D. committee, as well as **Prof. Dritan Nace** and **Dr. Luce Brotcorne**, who have kindly accepted the invitation to be reviewers of my Ph.D. thesis, for the time they spent reading my manuscript, and for their insightful comments that improved this thesis. My gratitude to **Dr. Boris Detienne** and **Prof. Safia Kedad-Sidhoum** for being

the examiner of my thesis and for their kind acceptance to take part in the jury of my Ph.D. defense.

I also want to thank warmly all the professors and colleagues from the **MAORE** team of Lirmm, **University of Montpellier**, the **university of Avignon** and also from the **university of Aveiro**. I also want to thank the administrative staff of Lirmm, in particular, **Nicolas Serrurier** for the kindness, availability, and efficiency have been an important help for me, allowing me to focus on my research.

This thesis has been financially supported by the Algerian ministry of higher education and scientific research and the consulate of Algeria in Montpellier. I would like to thank them for the funding received during my thesis.

I take this opportunity to thank my friends who helped to make these years such a nice experience. I am grateful that you have always been there whenever I needed you.

Last but not the least, I would like to thank my family: my parents, my brothers, and my sister for the support, encouragement and for believing in me.

# Résumé étendu

Au cours de ces dernières années, Les réseaux sont de plus en plus utilisés dans notre vie quotidienne. Des marchandises, des informations, des données et aussi des personnes doivent être transportées ou envoyées entre des différents endroits rapidement et à moindre coût. Par conséquent, ces réseaux doivent être conçus pour trouver la solution optimale (chemins, flux, sous-réseaux), en optimisant une fonction objective définie et en satisfaisant des exigences données.

La programmation mathématique est un outil pour trouver la meilleure solution de ces problèmes de conception de réseaux. Essentiellement, la programation mathématique modélise ces problèmes (i) en décidant les variables de décision, (ii) en définissant la fonction objective à optimiser (le but du problème) et (iii) en fixant les contraintes (exigences relatives aux solutions optimales). L'objectif et les contraintes sont caractérisés par des fonctions qui peuvent être linéaires, non linéaires, convexes ou non convexes. On peut également envisager d'optimiser une ou plusieurs fonctions objectives. Le problème peut considérer un ou plusieurs niveaux d'optimisation. Sur la base de toutes ces propriétés, on peut déterminer la complexité du problème et proposer le meilleur algorithme pour résoudre le problème efficacement.

Une classe importante de problèmes d'optimisation mathématique concerne les modèles de programmation linéaire à nombres entiers

mixtes (**MILP**), qui sont utilisés pour résoudre la plupart des problèmes de conception de réseaux. Dans cette classe, la fonction objective et les contraintes doivent être linéaires, et les variables du problème peuvent prendre des valeurs entières, binaires ou continues. En général, les méthodes d'énumération telles que les algorithmes de branch-and-bound et de branch-and-cut sont utilisées pour résoudre les problèmes (**MILP**).

Dans cette thèse, nous nous intéressons à un cas particulier de la programmation en nombres entiers où le problème est décomposé en deux niveaux de décision. Dans le premier niveau, le "leader" prend une décision en minimisant sa fonction objective. Ensuite, le problème du second niveau consiste à trouver la solution optimale du "follower" en fonction de la décision du "leader".

## Problématiques étudiées

La programmation linéaire en nombres entiers bi-niveaux a de nombreuses applications pour les problèmes de conception de réseaux. Dans cette thèse, trois applications principales sont étudiées. Pour chacune de ces applications, une modélisation mathématique et des algorithmes précis sont proposés pour résoudre le problème d'optimisation correspondant.

Commençons par présenter les applications étudiées pour les problèmes de conception de réseaux bi-niveaux dans cette thèse.

## Réseaux de transport de matières dangereuses

Dans ces dernières années, en raison de la croissance de l'industrie, d'énormes quantités de marchandises ont dû être transportées dans les centres urbains. Une grande partie de ces marchandises sont des matières dangereuses (Hazmats), qui représentent, par exemple, un envoi sur cinq sur les routes nord-américaines, et il y a au moins 800,000 envois routiers par jour [55]. En dépit de tous les records de sécurité pris par l'industrie, des accidents peuvent se produire. Pour réduire la possibilité et le coût des accidents, de nombreuses recherches ont été menées pour concevoir les réseaux routiers de matières dangereuses.

En considérant un réseau routier, et un ensemble de matières dangereuses à transporter entre les points d'origine et de destination. Le problème de la conception du réseau de matières dangereuses consiste à trouver une route pour chaque demande afin de minimiser le coût du transport. Dans cette thèse, nous considérons un modèle biniveaux où il y a deux agents pour prendre la décision. Au premier niveau, le gouvernement sélectionne un ensemble de routes à activer en minimisant le risque d'accidents. Ensuite, au second niveau, chaque transporteur choisit une route entre son origine et sa destination pour l'expédition de matières dangereuses associée avec un coût optimal. Le problème est formulé sous la forme d'un modèle de programmation linéaire en nombres entiers à deux niveaux.

## Réseaux de télécommunications

En raison de l'augmentation considérable du nombre d'appareils connectés aux réseaux IP, la consommation d'énergie de ces réseaux croît rapidement, avec un rythme de 10 % par an, ce qui représente 1,8 % de la consommation mondiale de l'électricité en 2012 [92]. En outre, on estime que les réseaux de communication consommeront jusqu'à 51 % de l'électricité mondiale dans le pire des cas d'ici 2030 si leur efficacité énergétique n'est pas suffisamment améliorée. Par conséquent, le problème de l'efficacité énergétique devient critique pour les réseaux de télécommunication d'aujourd'hui.

Afin de réduire la consommation d'énergie sur les réseaux IP, les réseaux durables ont attiré l'attention des fabricants d'appareils et des fournisseurs de services d'internet (ISP). Dans la littérature, un problème d'"energy-aware traffic engineering" est proposé pour minimiser la consommation totale d'énergie en éteignant les appareils de réseau inutilisés (routeurs et liaisons) tout en garantissant la connectivité du réseau.

Dans cette thèse, nous avons proposé un modèle d'optimisation à deux niveaux pour le problème de minimisation de la consommation d'énergie dans les réseaux ISP avec un protocole de routage multi-chemins pour minimiser l'utilisation de la capacité des liens. Dans ce problème, nous considérons un réseau ISP modélisé par un graphe bidirectionnel $G(V, A)$, où $V$ est l'ensemble des routeurs, et $A$ représente l'ensemble des liens de communication. Dans le premier niveau, nous sélectionnons un sous-ensemble d'appareils réseau à activer afin de minimiser la fonction de gestion de l'énergie. Ensuite,

au deuxième niveau, chaque demande est envoyée sur le réseau activé en satisfaisant un protocole de routage multi-chemins entre la paire origine-destination associée et en minimisant l'utilisation de la capacité des liens pour éviter la congestion dans le réseau.

## Réseaux sociaux

La dernière application du problème de conception de réseaux biniveaux étudiée dans cette thèse est le problème de maximisation d'influence dans les réseaux sociaux signés. Les réseaux sociaux peuvent être modélisés par des graphes représentant des individus et leurs relations, telles que amitié, collaboration ou les relations de recherche de conseils. Ces individus sont fréquemment influencés, explicitement ou implicitement, par leurs contacts sociaux.

Le nombre d'utilisateurs de réseaux sociaux a augmenté rapidement ces dernières années (environ 2,4 milliards d'utilisateurs pour Facebook, par exemple). En conséquence, la propagation de l'influence à travers les réseaux sociaux a une grande importance dans la décision d'adopter les informations (telle qu'une idée politique, un nouveau produit ou des innovations technologiques). L'étude de l'influence et les effets du "word of mouth" dans les réseaux sociaux est motivée par des applications telles que la diffusion d'idées ou d'innovations dans un réseau et le marketing viral de produits.

Les études actuelles se concentrent presque sur les réseaux sociaux non signés qui ne contiennent que des relations positives (par exemple, l'amitié ou la confiance) entre les utilisateurs. Dans cette thèse, nous étudions le problème de l'influence dans les réseaux so-

ciaux signés où le graphe contient à la fois des relations positives et négatives (par exemple, l'ennemi ou la méfiance) entre les utilisateurs. Le premier niveau du problème sélectionne les influenceurs, auxquels l'information est fournie directement. L'information est ensuite diffusée aux autres nœuds du réseau en utilisant le plus court chemin (qui est modélisé par le deuxième niveau), ce qui est conforme à la théorie *halo effect*.

## Plan de la thèse

Cette thèse est divisée en trois grands chapitres de contributions précédés d'un chapitre présentant les notions préliminaires. Ce chapitre commence par une présentation des problèmes de programmation linéaire en nombres entiers à un et à deux niveaux et une discussion des algorithmes utilisés pour résoudre ces problèmes. Ensuite, deux problèmes de flot utilisés dans les autres chapitres (problème du plus court chemin et du flot à coût minimum) sont introduits dans les deux dernières sections.

Le chapitre 3 est consacré à étudier et résoudre le problème de conception des réseaux à charges fixes avec des contraintes du plus court chemin par des algorithmes exacts. Nous proposons deux nouvelles formulations BILP basées respectivement sur des inégalités valides de chemin et de cycle utilisées pour éliminer les solutions non réalisables pour le problème bi-niveaux. La formulation du cycle est inspirée de la contribution dans [79]. Les relaxations linéaires des deux formulations sont comparées théoriquement.

Pour résoudre le problème étudié, nous proposons deux façons d'intégrer

les formulations de chemin et de cycle dans des algorithmes de Branch-and-cut et des algorithmes itératifs des plans sécants. Dans ces derniers, à chaque itération, la formulation avec une partie des contraintes du problème est résolue exactement, puis, un ensemble d'inégalités du plus court chemin est ajouté tant que la solution retournée est non réalisable. Ensuite, nous renforçons les formulations proposées par un ensemble d'inégalités valides qui s'appliquent dans le cas où différents produits ont la même origine et la même destination. Enfin, des tests numériques sont réalisés sur des ensembles de données réelles ainsi que sur des instances aléatoires. Ces travaux ont été publiés dans la revue de "Computers and Operation Research" [28], et ils sont présentés dans [24; 26].

Dans le chapitre 4, nous étudions le problème de la minimisation de la consommation d'énergie du réseau en utilisant un protocole de routage multi-chemins pour réduire l'utilisation de la capacité des liaisons.Nous formulons pour la première fois le problème étudié comme un problème d'optimisation bi-niveaux. Pour le résoudre, la formulation bi-niveaux est transformée en un modèle à un seul niveau en utilisant les conditions d'optimalité de KKT, des inégalités de flot à coût minimum et des conditions d'optimalité du graphe résiduel. Des méthodes itératives de "cutting plane" et de "branch-and-cut" sont ensuite proposées pour résoudre le problème. Ce chapitre est basé sur les travaux [25; 27].

Le chapitre 5 étudie le problème de maximisation d'influence dans les réseaux sociaux signés. Pour la première fois, un modèle de programmation bi-niveaux est proposé pour modéliser ce problème. Nous reformulons le problème en modèles MILP en utilisant trois condi-

tions d'optimalité différentes pour le problème du deuxième niveau. Ensuite, deux formulations sont proposées en utilisant des plus court chemins précalculés pour réduire le nombre de variables et de contraintes dans le modèle. Ces formulations à un niveau sont renforcées par un ensemble d'inégalités valides. Enfin, des cas particuliers de réseaux sociaux sont étudiés dans lesquels la solution optimale ou bornes inférieurs de la solution sont faciles à calculer. Des expérimentations sont réalisées en utilisant des instances aléatoires pour comparer les différentes formulations proposées.

La thèse se termine par un chapitre de conclusion (chapitre 6), présentant les résultats de la thèse et les perspectives dans ce domaine de recherche.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In the past years, we have been using more and more networks in our daily lives. Commodities, information, data, and people have to be transported or sent between different places quickly and cheaply. Consequently, these networks must be designed to find the best solution (paths, flows, sub-networks), optimizing a defined objective function and satisfying given requirements.

Mathematical optimization is a framework capable of finding the best solution to these network design problems. Essentially, mathematical optimization models the such problems by (i) deciding on which variables to optimize, (ii) defining the objective function to be optimized (the goal of the problem) and (iii) setting the constraints (requirements on the optimal solutions). The objective and the constraints are characterized by functions that can be linear, nonlinear, convex, or non-convex. Also, we can consider to optimize single or multiple objective functions. The problem can consider single or multiple levels of optimization. Based on all the above properties, we can determine the computational complexity of the problem and devise the best algorithm to solve the problem efficiently.

An important class of mathematical optimization problems concerns mixed-integer linear programming models (**MILP**), which are used to solve most of

network design problems. In this class, both of the objective function and the constraints are required to be linear, and the variables of the problem can take integer, binary, or continuous values. In general, enumeration methods like branch-and-bound and branch-and-cut algorithms are used to solve (**MILP**) problems.

In this thesis, we are interested in a particular case of integer programming where the problem is decomposed into two levels of decision. In the first level, the "leader" make a decision minimizing her objective function. Then, the second level problem consists in finding the optimal solution of the "follower" based on the decision of the leader.

## 1.1 Bi-level network design problem applications

Bi-level integer programming has many applications for network design problems. In this thesis, three main applications are studied. For each of these applications, mathematical modeling and exact algorithms are proposed to solve the corresponding optimization problem. Let us start by presenting the studied applications for bi-level network design problem along this thesis.

### 1.1.1 Hazmat transportation networks

In the last years, because of the growth of industry, huge amounts of commodities have to be transported in urban centers. A big part of these commodities are hazardous materials (Hazmats), which represent, for example, one in five shipments on North American highways, and there are at least 800,000 road shipments per day [55]. In spite of all safety records taken on industry, accidents may happen. To reduce the possibility and the costs of accidents, a lot of research has been carried out to design the road networks of hazardous materials.

Considering a road network, and a set of hazmats to be transported between

origin and destination points. The hazmat network design problem consists in designing a road for each commodity in order to minimize the transportation cost. In this thesis, we consider a bi-level model where there are two agents to make the decision. In the first level, the government selects a set of roads to be activated, minimizing the risk of accidents. Then, in the second level, each carrier selects a route between the origin and the destination for the associated hazmat shipment with an optimum cost. The problem is formulated as a bi-level integer programming model.

### 1.1.2 Telecommunication networks

Due to the huge growth of the number of devices connected to IP networks, the network energy consumption is inherently growing fast with a rate of 10% per year, which represented a 1.8% of the worldwide electricity consumption in 2012 [92]. Furthermore, communication networks are estimated to consume as much as 51% of the global electricity in the worst case by 2030 if their energy efficiency is not enhanced enough [8]. Therefore, the problem of energy efficiency is becoming critical for nowadays communication networks.

To reduce energy consumption on IP networks, green networking has attracted much attention from device manufacturers and Internet Service Providers (ISP). In the literature, an energy-aware traffic engineering problem is proposed to minimize the total energy consumption by switching off unused network devices (routers and links) while guaranteeing full network connectivity.

In this thesis, we proposed a bi-level optimization model for the problem of energy-aware Traffic Engineering (TE) while using multi-path routing to minimize link capacity utilization in ISP backbone networks. In this problem, we consider an ISP backbone network modeled by the bi-directed graph $G(V, A)$, where $V$ is the set of routers, and $A$ represents the set of communication links. In the first

level, we select a subset of network devices to be powered on in order to minimize the energy management function. Then, in the second level, each demand is sent on the activated network satisfying a multi-path routing protocol between the associated pair of origin-destination and minimizing the total link capacity utilization to avoid congestion in the network.

### 1.1.3 Social networks

The last application of the bi-level network design problem studied in this thesis is the maximum influence problem in signed social networks. Social networks can be modeled by graphs representing individuals and their relationships, such as friendships, collaborations, or advice-seeking relationships. These individuals are frequently influenced, explicitly, or implicitly by their social contacts.

The number of social networks users has been increasing rapidly in these last years (about 2.4 billion users for Facebook, for instance). In consequence, influence propagation through social network has a great importance in deciding whether to adopt an innovation (such as a political idea, a new product, or technological innovations). Studying the influence and the effects of the "word of mouth" in the promotion of new products in social networks is motivated by applications like the spread of ideas or innovations in a network and viral marketing of products.

Current studies focus almost on unsigned social networks containing only positive relationships (e.g., friendship or trust) between users. In this thesis, we study the influence problem in signed social networks where the graph contains both positive and negative relationships (e.g., foe or distrust) between users. The first level of the problem selects the influencers, to which the information is provided directly. The information is then spread to the other nodes of the network using shortest path (which is modeled by the second level), which is line

with the *halo effect* theory.

## 1.2 Outline of the Thesis

This thesis is divided into three main contribution chapters preceded by a chapter introducing preliminaries notions. This chapter starts with a presentation of the one-level and bi-level integer programming problems and a discussion of the algorithms used to solve these problems. Then, two network flow problems used in the other chapters (shortest path and minimum cost flow problems) are introduced in the two last subsections.

Chapter 3 is devoted to studying and solving the fixed-charge network design problem with shortest path constraints by exact algorithms. We propose two new BILP formulations based, respectively, on path and cycle valid inequalities used to eliminate the infeasible bi-level solutions. The cycle formulation is inspired by the contribution in [79]. The linear relaxations of the two BILP formulations are compared theoretically. To solve the studied problem, we propose two ways of integrating the path and cycle formulations in cutting plane algorithms, using either a branch-and-cut or an iterative cutting-plane strategy. In the second strategy, at each iteration, a partial ILP formulation of FCNDP-SPC is solved exactly, and a set of shortest path inequalities is added while the returned solution is unfeasible. Then, we strengthen our formulations through a set of valid inequalities that apply to the case where different commodities have the same origin and the same destination. Finally, numerical experiments are done on real data sets from the literature as well as on random instances. This work has been published in Computers and Operation research journal [28], and it is presented in [24; 26].

In Chapter 4, we study the problem of minimizing the network energy con-

sumption while using multi-path routing to reduce link capacity utilization. To the best of our knowledge, it is the first time that this problem is investigated. From the perspective of optimization techniques, we formulate the studied problem as a bi-level optimization for the first time. To solve it, the bi-level formulation is transformed into a one-level optimization using KKT conditions, minimum flow inequalities or residual network optimality conditions. Iterative cutting plane and branch-and-cut methods are further proposed to solve the problem. This chapter is based on works [25; 27].

Chapter 5 studies the maximum influence problem in signed social networks. For the first time, a bi-level programming model is proposed to model the problem. We reformulate the two-levels problem into one-level MILP models using three different optimality conditions for the second-level problem. Then, two formulations are proposed using pre-calculated shortest path distances to reduce the number of variables and constraints in the model. These one-level formulations are strengthened by adding a set of valid inequalities. Finally, special cases of the networks in which the optimal solution or bounds are easy to find are studied. Computational experiments are performed using random and real instances to compare the different proposed formulations.

The thesis ends with a concluding chapter (Chapter 6), presenting the results of the thesis and perspectives in this research field.

# Chapter 2

# Preliminary Notions

This chapter is devoted to the introduction of the basic optimization problems addressed in this thesis. Precisely, integer and bi-level programming are presented in the first two subsections. In the last two subsections, we present two network flow problems: shortest path and minimum cost flow problem. For each problem, we give the definition, some fundamental theoretical properties, mathematical formulations, and the well-known algorithms used to solve these problems.

## 2.1 Integer programming

Mixed-integer linear programming (abbreviated MILP in what follows) denotes constrained optimization problems in which the objective function and all constraints are linear, and some or all decision variables are restricted to be integer. In the later case the terminology ILP is used instead of MILP. MILP can be used to formulate and solve most combinatorial optimization problems, and it has a wide range of applications, such as transportation, scheduling, and telecommunication networks.

### 2.1.1 Definition of MILP problem

The integer linear programming problem can be written in the general form as follows [141]:

$$(ILP)\begin{cases}\min & cx \\ \text{s.t.} & Ax \leq b, \\ & x \geq 0, \quad x \in \mathbb{Z}^n,\end{cases}$$

where $A$ is an integer $m \times n$ matrix and $b$ and $c$ are vectors of dimensions $m$ and $n$, respectively. The optimal solution of ILP is given by the integer vector $x = (x_1, ..., x_n)^T$. The problem is called a mixed-integer linear program (MILP) if some variables are not required to be integer. This formulation can include equality constraints because each of them can be replaced by two inequality constraints.

The linear relaxation of ILP is the linear optimization problem obtained by relaxing the integrity constraints $x \in \mathbb{Z}^n$, and it is given by:

$$(LP)\begin{cases}\min & cx \\ \text{s.t.} & Ax \leq b, \\ & x \geq 0.\end{cases}$$

(LP) is an easy optimization problem as it can be solved in polynomial time. As (LP) and (ILP) have a constrictive relation, and since (LP) is easy to solve, (LP) can be used as an attempt to solve the (ILP). If it happens that the optimal solution of linear relaxation (LP) is integral, then (ILP) is easy to solve.

### 2.1.2 Complexity

In what follows, we provide a few simple notions of the complexity theory of optimization problems (more detailed in [82; 141]).

Defining the complexity of a combinatorial optimization problem $(P)$ requires first introducing the associated decision problem $(D)$. Problem $(D) \in \mathcal{NP}$ if we can check in polynomial time whether a candidate solution provides an answer to (D) (more details about polynomial time complexity in [82]).

If every problem in the class $\mathcal{NP}$ can be polynomially reduced to $(D)$, then, $(D)$ is $\mathcal{NP}$-complete, and its associated optimization problem (P) is $\mathcal{NP}-$hard [141]. Moreover, an optimization problem $(P)$ is polynomially solvable and $(P) \in \mathcal{P}$ if there exists an algorithm to solve the worst-case of the problem in polynomial time [122].

The mixed-integer linear programming problems belong to the class of $\mathcal{NP}$-hard problems. This result can be verified by the reduction from $\mathcal{NP}$-hard problems (for example, the vertex cover problem).

### 2.1.3 Exact algorithms to solve MILP

Developing exact algorithms to solve ILPs and MILPs has been a subject of many studies in the last 50 years. The proposed algorithms belong to three categories of methods depending on the type of the used method to get the optimal solution [66]:

**Polyhedral approaches:** cutting plane algorithm

**Enumerative approaches:** branch-and-bound, branch-and-cut, branch-and-price

**Relaxation and decomposition techniques:** lagrangian relaxation, Benders decomposition.

In this subsection, we will give the basic ideas for some of the mentioned algorithms (the ones used in this thesis).

**Branch-and-Bound** Branch-and-Bound algorithm is one of the most studied algorithms to solve $\mathcal{NP}$-hard combinatorial optimization algorithms. Generally, these problems have exponential complexity, and solution algorithms need to enumerate a large number of feasible solutions to find the optimal one. The branch-and-bound method was proposed the first time in 1964 by Bertier and Roy [19], and it has been studied by many researchers [3; 12; 93; 123].

Branch-and-bound algorithms start by relaxing the integrity constraints, and solving the linear relaxation of the problem. If the obtained solution is integer, then this solution is optimal for the integer problem. Otherwise, there must exist at least one variable $x_j$ with a fractional value $\alpha$. The idea of the algorithm is to separate the problem into two sub-problems (branching). The first sub-problem contains the constraint $x_j \geq \lceil \alpha \rceil$ and the second includes the constraint $x_j \leq \lceil \alpha \rceil - 1$. It is clear that this creates a partition of the feasible set of the original integer problem. The process is repeated for each sub-problem.

This procedure is represented as a search tree, where at each level, a partition of the parent vertex is performed according to the rule described above. It is then a question of going through this enumeration tree in order to find the optimal solution. Exploring a tree path can end (bounding) for three reasons:

- The solution of the sub-problem is integer.
- The sub-problem becomes unfeasible.
- The optimal solution of the sub-problem is worst than a feasible solution encountered so far.

The optimal solution of the initial problem is the best solution reached in the branch-and-bound tree.

**Cutting plane** The cutting plane algorithm is an exact iterative method to solve mixed-integer programming problems [83]. It has also been used to solve many combinatorial problems including packing problems [77; 116], traveling salesman problem [9; 10; 119] and the vehicle routing problem [45; 129].

The cutting plane algorithm has been proposed for general ILPs for the first time by R.E. Gomory [69; 70]. The basic idea of the algorithm is to solve at each iteration the linear relaxation of the problem. If the obtained solution is integer, then it is optimal for the ILP problem. Otherwise, we add a constraint that eliminates this solution. Then, the basic structure of the cutting plane algorithm is given by Algorithm 1.

**Algorithm 1** Cutting plane algorithm

**Step 1:** Solve the linear relaxation of the initial problem.

**Step 2:** If the solution of the linear relaxation is integer, then, it is optimal.

**Step 3:** Otherwise, generate a cutting plane (valid inequality) that separates the obtained solution (continuous) from the convex hull of integral solutions of ILP.

**Step 4:** Return to **step 1**.

The efficiency of the cutting plane algorithm depends on the strength of the generated valid inequalities at each iteration. Its limited efficiency at the time of its introduction and in subsequent years explains that this method has been neglected for many years until the development of polyhedral theory in the eighties. Nowadays, the cutting plane algorithm is more efficient and used to solve a variety of combinatorial problems.

**Branch-and-Cut** Since ILPs and MILPs are $\mathcal{NP}-$hard, finding an optimal solution in a reasonable time with the presented algorithms can be improbable. As a consequence, combining many algorithms can improve the solution time.

Branch-and-cut algorithm is an exact method to solve linear integer problems that combines branch-and-bound with the cutting plane algorithm. This method consists in solving the problem using a branch-and-bound procedure and use the cutting plane method to eliminate unfeasible solution in each node of the branch-and-bound tree.

The advantage of this algorithm is to reduce the number of nodes of the branch-and-bound tree by improving the strength of the formulation dynamically in the course of the branch-and-bound tree.

## 2.2 Bi-level programming

Bi-level programming is a hierarchical mathematical optimization problem where the decision is made by two independent and non-cooperative agents. The first one is called a "leader", whereas the second one the "follower". Each of them has her own objective. The agents act and react in a sequential and interdependent way [21; 48; 121]. In our case, the objective of the second level problem is to minimize her objective linear function according to the first level decision. On this assumption, the leader also specifies a decision such that the objective function of the upper-level problem is minimized. The obtained solution applying the above procedure is called an equilibrium solution, which defines the optimal solution of the bi-level optimization problem [18; 20; 32; 108; 117].

From a historical point of view, one of the first applications of two-level programming was in the economic problem of the Stackelberg game [135]. This application represents an economical scenario in which the leader and the follower

are two agents (firms or individuals) who operate in the same market. The objective of both of them is to maximize their profits. Several other applications can be found for bi-level optimization, for example in network design, transportation networks or game theory [29; 30; 112; 120].

### 2.2.1 Linear bi-level programming

The majority of carried out studies on bi-level programming has focused on linear versions. The mathematical formulation of the linear bi-level programming problem (LBP) is the following:

$$\min_{x \in X} \quad c_1 x + d_1 y \tag{2.1a}$$

$$\text{s.t.} \quad A_1 x + B_1 y \leq b_1, \tag{2.1b}$$

$$\min_{y \in Y} \quad c_2 x + d_2 y \tag{2.1c}$$

$$\text{s.t.} \quad A_2 x + B_2 y \leq b_2 \tag{2.1d}$$

where $c_1, d_1, c_2, d_2$ are the objective functions vectors, and $A_1, B_1, A_2, B_2$ represent constraints matrices. The variable $x$ is called the "upper level" variable, and $y$ is called the "lower level" variable. Both variables are defined on the feasible regions $X$ and $Y$ respectively.

The linear bi-level problem is $\mathcal{NP}-$hard even if the first and the second level problems are continuous linear problems [44].

To analyze the problem, let us define the following sets [121]:

- The feasible region of (LBP) is:

$$S = \{(x, y) \in X \times Y : \ A_1 x + B_1 y \leq b_1, \ A_2 x + B_2 y \leq b_2\}$$

- The feasible region of the "follower" for a given value of $x$ is:

$$S(x) = \{y \in Y : \quad A_2x + B_2y \leq b_2\}$$

- The inducible region of the bi-level problem is given by:

$$IR = \{(x, y) \in S : y = \arg\min_{\bar{y} \in Y} \{ \ c_2x + d_2\bar{y}, \quad \bar{y} \in S(x)\}\}$$

- A solution $(x^*, y^*) \in X \times Y$ is bi-level optimal if, and only if:

$$c_1x^* + d_1y^* \leq c_1x + d_1y, \quad \forall (x, y) \in IR.$$

### 2.2.2 Classes of bi-level programming problems

There exist three different classes of linear bi-level programming problems according to the type of decision variables $x$ and $y$ [134]. These classes are defined as:

**Continuous linear bi-level programming CLBP:** The decision variables of both first and second level problems in this class are continuous $x, y \in \mathbb{R}^n$. CLBP is the simplest case of bi-level optimization problem even if it belongs to the class of $\mathcal{NP}-$hard problems. Many applications can be found for CLBP especially when the two levels problems do not need discrete variables, for instance continuous transportation problems, continuous network design problems and many other applications [40; 58; 136].

**Discrete linear bi-level programming DLBP:** The decision variables in this class are both discrete $x, y \in \mathbb{Z}^n$ and if the first level variable is continuous this class called continuous-discrete BLBP CDLBP. Therefore, the bi-level model

cannot be solved reformulating into one-level problem. The DLBP and CDLBP classes have many applications in transportation, industry as well as telecommunication, and it can be solved only by enumeration approaches [14; 47; 113; 134].

**Discrete-Continuous linear bi-level programming DCLBP:** In this class, the decision variable of the leader is discrete $x \in \mathbb{Z}^n$ while the second level is a continuous problem $y \in \mathbb{R}^n$. This is a generalization of CLBP and can be solved by the same algorithms. DCLBP has appeared in many application (Hazardous transportation, facility location, and network design problems) and many methods are proposed to solve the problem in [5; 21; 22; 28; 73; 84; 110].

### 2.2.3 Non-unique follower optimal solution

In the definition of the inducible region of LBP, we have seen that the feasible solutions $(x, y)$ depend on the optimal solution of the follower problem reacting to $x$. Since we have an optimization problem in the second level problem, the follower may have multiple optimal solutions for the same value of the leader variable $x$.

Note that in bi-level optimization problems, the leader cannot interfere with the follower's decision to force her to choose one among the multiple solutions. Hence, the leader cannot predict the chosen follower's optimal solution. Thus, we distinguish two scenarios of LBP depending on the optimal solution of the follower [47]:

**The optimistic position:** The first scenario is the optimistic position. In this case, we suppose that the follower will take the best solution for the leader from her multiple optimal solutions. Most of the contributions on bi-level programming address this scenario.

**The pessimistic position:** The optimistic position is not possible for all problems, especially in real-world applications where the cooperation between the leader and the follower is not allowed, or it is not possible to know the decision of the follower. In this case, the problem will be solved with the assumption that follower will choose the worst solution for the leader objective.

In addition to these two scenarios, researchers have also proposed heuristic algorithms that select follower solutions without considering their impact into the leader's objective, e.g. [132].

### 2.2.4 Algorithms to solve LBP

In general, three types of algorithms are used to solve linear programming problems [121]:

**Solutions enumeration algorithm:** Since the inducible region of LBP is defined by the finite elements set of bi-level feasible solutions (with the assumption that $IR$ is bounded):

$$IR = \{(x, y) \in S : y = \arg\min_{\bar{y} \in Y} \{c_2 x + d_2 \bar{y}, \quad \bar{y} \in S(x)\}\}$$

the optimal solution can obtained by enumerating all feasible solutions and taking the best one for the leader objective function.

**KKT approach:** In the case of a continuous lower level problem (CLBP and DCLBP), the bi-level problem can be reformulated as a single level MILP problem replacing the second level problem by the associated KKT optimality conditions. Thus, the problem can be solved by any integer programming algorithm.

Let us consider the discrete-continuous version of the bi-level linear model

(2.1) presented in Subsection 2.2.1:

$$\min_{x \in \mathbb{Z}^n} \quad c_1 x + d_1 y \tag{2.2a}$$

$$\text{s.t.} \quad A_1 x + B_1 y \leq b_1, \tag{2.2b}$$

$$\min_{y \in \mathbb{R}^n} \quad c_2 x + d_2 y \tag{2.2c}$$

$$\text{s.t.} \quad A_2 x + B_2 y \leq b_2. \tag{2.2d}$$

The second level problem can be replaced by the associated KKT optimality condition, then, the problem can be reformulated as:

$$\min_{x \in \mathbb{Z}^n} \quad c_1 x + d_1 y \tag{2.3a}$$

$$\text{s.t.} \quad A_1 x + B_1 y \leq b_1, \tag{2.3b}$$

$$A_2 x + B_2 y \leq b_2, \tag{2.3c}$$

$$\langle y, (d_2 + \lambda^T B_2) \rangle = 0, \tag{2.3d}$$

$$\langle \lambda, A_2 x + B_2 y - b_2 \rangle = 0, \tag{2.3e}$$

$$\lambda \in \mathbb{R}^{m+}. \tag{2.3f}$$

This last formulation contains non-linear constraints (2.3d) and (2.3e) (the symbol $\langle ., . \rangle$ represents the scalar product between two vectors), which can be linearized by adding binary variables and Big-M method to get a MILP problem solvable with integer programming algorithms, presented in Section 2.1.

In this thesis, the studied problems belong to the class of discrete-continuous linear bi-level programming problems, and we consider the optimistic position in the case of multiple optimal solutions of the second level problem.

**Penalty approach:** The basic idea of this algorithm is to convert the follower problem to an unconstrained minimization problem using a barrier method.

Then, a penalty on the dual gap of the second level problem will be introduced alternatively on the objective function of the leader.

## 2.3 Shortest path problem

In operations research, the shortest path problem is one of the most fundamental and simplest problems which has many applications in practice and theory. Among these applications, we find telecommunication routing protocols, traffic management, transportation planning, and social networks, and so on. For a given weighted directed or undirected graph $G(V, E)$ with a set of $n$ nodes $V$, a set of $m$ edges $E$ and a positive cost $c_e$ associated to each edge $e \in E$, let $A$ be the set of directed arcs associated to $E$. The classical shortest path problem looks for a path $P$ with minimum cost between a source node $s \in V$ and a destination node $d \in V$ [4; 105].

This is the basic version of the shortest path problem. In the literature, many generalizations are studied, for instance:

- The shortest path from one single source node to all the nodes of the graph.
- The shortest path from all the nodes of the graph to one destination node.
- The shortest path between all pairs of vertices in the graph.

### 2.3.1 MILP formulation

The single-source single-destination shortest path problem can be formulated as a linear programming model. We are given an undirected graph $G(V, E)$ with a cost $c_{ij}$ for each arc $(i, j) \in \{(u, v), (v, u) : \{u, v\} \in E\}$. The problem to find the

shortest path between the two nodes $s$ and $d$ can be formulated as:

$$\min \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.4a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in\delta^+(i)} x_{ij} - \sum_{(i,j)\in\delta^-(i)} x_{ji} = \begin{cases} 1, & \text{if } i = s, \\ -1, & \text{if } i = d, \\ 0, & \text{otherwise.} \end{cases} \qquad \forall i \in V, \tag{2.4b}$$

$$x_{ij} \geq 0, \qquad \forall (i,j) \in A. \tag{2.4c}$$

The objective function (2.4a) minimizes the total length of the path defined by the positive decision variable $x$ which indicates that the arc $(i, j)$ is in the optimal solution if $x_{ij} = 1$. Finally, constraint (2.4b) ensures to find only one path between $s$ and $d$. This LP formulation has a particular property that its optimal solution is integral.

**Theorem 1** (Existence, [89])**.** *The shortest path problem between the two nodes $s$ and $d$ in the graph $G(V, E)$ has a solution if and only if $G$ has no negative cycle.*

### 2.3.2 Algorithms

The shortest path problem is one of the easiest combinatorial optimization problems to solve due to its linear formulation. In this part, we describe some fundamental algorithms to solve the problem.

**Dijkstra algorithm:** Dijkstra algorithm is a graph search method that finds the shortest path between nodes in non-negative weighted graphs. This method is proposed by Edsger Dijkstra in 1956 and published three years after [52].

There exist many variations of the algorithm, the original one finds in either

oriented or not-oriented graph the shortest path between a fixed source node $s$ and all graph vertices producing a shortest path tree.

To find a single-source single-destination shortest path, it is sufficient to stop once the length of the shortest path to the destination node is determined by the algorithm.

This algorithm is part of the label setting algorithms family. The main idea of Dijkstra's original algorithm (more detailed in Algorithm 2) is to select, at each iteration, a vertex $u$ calculate the label's final value $\pi(u)$, the length of the shortest path from $s$ to $u$ (see also: [71; 90; 125]).

**Algorithm 2** Dijkstra algorithm

**Input:** Graph **G(V,E)**, source node **s** and weights $c_e \geq 0, \quad \forall e \in E$.

**Initialization:** $\pi(u) = \infty$, for all $u \in V \backslash \{s\}$ and $\pi(s) = 0$. $\quad Q = \emptyset$.

**Iterations:**

**Step 1:** Find a vertex $u \in V \backslash Q$ such that: $\pi(u) = \min_{v \in V \backslash Q} \pi(v)$.

**Step 2:** Set: $Q \leftarrow Q \cup \{u\}$.

**Step 3:** For all node $v \in V \backslash Q$ neighbor of $u$ do:
If $\pi(v) \geq \pi(u) + c_{uv}$ then:
set: $\pi(v) \leftarrow \pi(u) + c_{uv}$.

**Step 4:** If $Q \neq V$, then go to **step 1**.

**Output:** $\pi(u)$ is the length of the shortest path between $s$ and $u$.

**Theorem 2** (Dijkstra [52])**.** *DIJKSTRA'S ALGORITHM works correctly.*

*Proof.* See [90]. □

**Theorem 3** (Fredman and Tarjan [64])**.** *The running time of DIJKSTRA'S ALGORITHM implemented with a Fibonacci heap is* $O(m + n \log n)$.

*Proof.* See [90]. □

**Bellman-Ford algorithm:** Dijkstra algorithm cannot be used in the case of a graph with negative arc weights. For the general case, Richard Bellman and Lester Ford proposed in 1958 a new algorithm, called Bellman-Ford algorithm to solve the shortest path from a source node $s$ to all other nodes in a weighted graph. Moreover, the algorithm indicates if the graph contains a negative-weight cycle, in which case the problem has no solution. This algorithm is based on the optimality conditions recalled below.

**Theorem 4** (Bellman's optimality conditions)**.** *$z$ is the length of the shortest path between $s$ and $d$ if, and only if, there exists a value $\pi_i$ for all $i \in V$ such that:*

- $\pi_s = 0$.
- $\pi_d = z$.
- $\pi_j \leq \pi_i + c_{ij}, \qquad \forall (i,j) \in A$.

*Proof.* See [141]. □

The Bellman-Ford algorithm is presented in Algorithm 3.

**Algorithm 3** Bellman-Ford algorithm [17; 62]

**Input:** Graph **G(V,E)**, source node $s$ and weights $c_e \in \mathbb{R}, \quad \forall e \in E$.

**Initialization:** $\pi(u) = \infty,\ p(u) =$Null for all $u \in V \backslash \{s\}$. $\quad \pi(s) = 0$.

**Iterations:**

**For** $i \leftarrow 1$ to $n-1$ **do:**

**For each** $(u, v) \in E$ **do**:

**If:** $\pi(v) > \pi(u) + c_{uv}$ **do**:

**Set:** $\pi(v) \leftarrow \pi(u) + c_{uv}$.

$p(v) \leftarrow u$.

**For each edge:**$(u, v) \in E$ **do:**

**If** $\pi(u) + c_{uv} > \pi(v)$ :

**Error:** $G$ contains a negative-weight cycle.

**Output:** $\pi(u)$ is the length of the shortest path between $s$ and $u$ and $p(u)$ is the predecessor of $u$ in the shortest path.

**Theorem 5** (Bellman [17], Ford [62])**.** *Bellman-Ford ALGORITHM works correctly with the running time $O(mn)$.*

*Proof.* See [90]. □

**All-pairs shortest path problem:** Suppose now that we want to solve the shortest path problem between all pair of nodes $(s, d) \in V$ in the weighted graph $G(V, E)$ with $c_e \in \mathbb{R},\ \forall e \in E$. It is clear that we can use the Bellman-Ford algorithm for each source node $s \in V$. The algorithm will be repeated $n$ times, which give us a running time $O(n^2 m)$.

In 1962, Robert Floyd and Stephen Warshall proposed an $O(n^3)$-algorithm (Algorithm 4) to find the shortest path between all pairs of nodes in the graph.

**Algorithm 4** Floyd-Warshall algorithm [61; 140]

**Input:** Weighted graph **G(V,E)** with $c_e \in \mathbb{R}, \quad \forall e \in E$.

**Initialization:** $\pi(u,v) = c_{uv}$, for all $(u,v) \in E$

$\pi(u,v) = \infty$, for all $(u,v) \in (V \times V) \backslash E, \ u \neq v$.

$\pi(u,u) = 0$, for all $u \in V$.

$p(u,v) = u$, for all $u, v \in V$.

**Iterations:**

**For** $v \leftarrow 1$ to $n$ **do:**

**For** $u \leftarrow 1$ to $n$, **if:** $u \neq v$ **do**:

**For** $w \leftarrow 1$ to $n$, **if:** $w \neq v$ **do**:

**If:** $\pi(u,w) > \pi(u,v) + \pi(v,w)$, **then:**

$\pi(u,w) \leftarrow \pi(u,v) + \pi(v,w)$.

$p(u,w) \leftarrow \pi(v,w)$.

**Output:** $\pi(u,v)$ is the length of the shortest path between the two nodes **u** and **v** and $(p(u,v),v)$ are the edges of the shortest paths to $j$ if they exist.

**Theorem 6** (Floyd [61], Warshall [140])**.** *The Floyd-Warshall ALGORITHM finds the shortest path between all pairs of nodes in a running time* $O(n^3)$.

*Proof.* See [90]. □

**Remark 1.** *Since the shortest path problem is formulated as a linear program, linear programming methodologies can also be used to solve the problem.*

## 2.4 Minimum cost flow problem

Minimum cost flow is a generalization of the shortest path problem in which the arcs in the network are associated with costs and capacities. The problem

consists in finding the best way to send an amount of flow between a source and a destination nodes in the network while minimizing the total cost. In this subsection, we will give the definition of the one-source one-destination variant, optimality conditions and some algorithms to solve the problem.

### 2.4.1 Definition

Given a directed network $G(V, A)$ supposed to be connected. We associate to each arc in the graph a cost $c_{ij}$ and a capacity $C_{ij}$. Assuming that we have a flow $\phi$ to be sent from a source node $s$ to a destination node $d$. The optimal solution of the minimum cost problem is a flow between the source and destination vertices, which satisfies the capacity constraint and minimizes the total cost.

The problem can be formulated as a linear program:

$$\min \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.5a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in\delta^+(i)} x_{ij} - \sum_{(i,j)\in\delta^-(i)} x_{ji} = \begin{cases} \phi, & \text{if } i = s, \\ -\phi, & \text{if } i = d, \\ 0, & \text{otherwise.} \end{cases}, \qquad \forall i \in V, \tag{2.5b}$$

$$x_{ij} \leq C_{ij}, \qquad \forall (i,j) \in A, \tag{2.5c}$$

$$x_{ij} \geq 0, \qquad \forall (i,j) \in A. \tag{2.5d}$$

The objective function (2.5a) minimizes the cost of the flow defined by the positive decision variable $x$ (2.4c), which indicates the amount of flow passed through the arc $(i, j)$. Finally, Constraints (2.5b), (2.5c) represent flow conservation and capacity constraints, respectively.

### 2.4.2 Optimality conditions

In this part, we discuss some theoretical properties of the minimum cost flow problem, in particular the optimality conditions, which will be used in Chapter 4.

We will start by defining the concept of residual networks.

**Definition 1** (Residual network)**.** *The residual network associated to the flow defined by the variable $x \in \mathbb{R}_+^{|A|}$ is the graph denoted by $G(x) = (V, A(x))$ and defined by the set of activated and unsaturated arcs $A(x)$. This set of arcs is defined as the union of the two following sets:*

$$A^+(x) = \{(i,j) \in A : 0 < x_{ij} < C_{ij}, \quad x_{ji} = 0\}$$
$$A^-(x) = \{(i,j) \in A : \qquad x_{ji} > 0\}.$$

*Each arc of $G(x)$ is associated with a residual cost $c'$ defined as:*

$$c'_{ij} = \begin{cases} c_{ij}, & \text{if } (i,j) \in A^+, \\ -c_{ij}, & \text{if } (i,j) \in A^-. \end{cases}$$

**Theorem 7** (Negative cycle optimality condition, [4])**.** *A feasible flow $x$ is optimal if and only if the network $G(x)$ has no negative cycles.*

*Proof.* See, [4] □

We provide now another form of negative cycle optimality conditions based on reduced costs.

**Theorem 8** (Reduced cost optimality condition [4])**.** *A feasible solution $x$ to* (2.5) *is optimal for the minimum cost flow problem if and only if there exist node*

*potentials* $\Pi$ *that satisfy the reduced cost optimality conditions:*

$$c'_{ij} + \Pi_i - \Pi_j \geq 0, \qquad \forall (i,j) \in G(x). \tag{2.6a}$$

Theorems 7 and 8 are applied on the residual network. In contrast, we present next these conditions for the initial graph $G$.

**Theorem 9** (Complementary slackness optimality conditions [4])**.** *A feasible solution $x$ is optimal for the minimum cost flow problem if and only if there exist node potentials* $\Pi$ *and reduced costs that satisfy the following complementary slackness optimality conditions:*

$$\textit{If} \quad c'_{ij} + \Pi_i - \Pi_j > 0, \qquad \textit{then,} \quad x_{ij} = 0, \tag{2.7a}$$

$$\textit{If} \quad 0 < x_{ij} < C_{ij}, \qquad \textit{then,} \quad c'_{ij} + \Pi_i - \Pi_j = 0, \tag{2.7b}$$

$$\textit{If} \quad c'_{ij} + \Pi_i - \Pi_j < 0, \qquad \textit{then,} \quad x_{ij} = C_{ij}. \tag{2.7c}$$

We can remark that these conditions are the complementary slackness optimality conditions of the linear formulation presented in Subsection 2.4.1.

### 2.4.3 Solution methods

The minimum cost flow problem can be solved with linear programming algorithms using the linear formulation. Besides this, in literature, many combinatorial algorithms have been proposed to solve the problem based on the optimality conditions presented before, or using a generalized algorithms of other optimization problems (shortest path, maximum flow,...). In this subsection, we present some of these algorithms.

**Cycle-canceling algorithm:** The negative cycle optimality conditions (Theorem 7) can be used to solve the minimum cost flow problem with an iterative approach. This algorithm preserves a flow $x$, tries to test the existence of negative cycles in the residual network at each iteration, and eliminates these cycles. The algorithm ends when the residual network $G(x)$ does not contain any negative cycles [4].

**Algorithm 5** Cycle-canceling algorithm

**Step 0:** Make a feasible flow $x'$.

**Step 1: While:** $G(x')$ contains a negative cycle **do**:

- Find a negative cycle $\mathcal{C}_{x'}$ in $G(x')$.
- Let $r$ be the minimum residual capacity of an edge on $\mathcal{C}$

$$r = \min_{(i,j)\in\mathcal{C}_{x'}} C_{ij} - x'_{ij}.$$

- Augment $r$ units of flow on the cycle $\mathcal{C}_{x'}$ to find new value of the solution $x'$.
- Update the residual network $G(x')$.

**Successive shortest path algorithm:** The minimum cost flow problem is a generalization of the shortest path problem, the only difference is the existence of capacity constraint. Thus, the two problems are equivalent in the case of all edge capacities are sufficient to pass the flow $\phi$.

Based on this, the problem can be solved iteratively using shortest path algorithm as follows:

**Algorithm 6** Successive shortest path algorithm

**Initialization:** Let $\Phi = 0$.

**While:** $\Phi < \phi$ **do**:

- Find the shortest path $p$ between $s$ and $d$.
- Let $r$ be the minimum arc capacity in $p$.
- **If** $r \geq \phi$, **then,** $p$ is the optimal solution
- **Else:** set $\Phi \leftarrow \Phi + r$.

# Chapter 3

# Fixed Charge Network Design Problem with Shortest-path constraints

In this chapter, we study the fixed charge network design problem with shortest path constraints which is modeled as a bi-level program. We first review three one-level formulations obtained by applying the complementarity slackness theorem, Bellman's optimality conditions and cycle elimination constraints. We propose two new binary integer programming (BILP) formulations inspired by path and cycle inequalities. The two formulations have exponential numbers of constraints. We incorporate the path and the cycle based formulations in a branch-and-cut algorithm and in another cutting-plane based method. Numerical experiments are performed on real instances, and random data sets generated with different criteria to examine the difficulty of the instances. The results show that the proposed cutting plane algorithms can solve up to 19% more instances than the classic branch-and-bound algorithms.

## 3.1 Introduction

The fixed charge network design problem (FCNDP) consists of selecting a subset of edges from a given network, in such a way that a set of commodities can be transported from its origins to its destinations. The objective is to minimize the sum of fixed costs (depending on selected edges) and variable costs (depending on the flow of commodities on the edges). In general, fixed and variable costs can be represented by linear functions, and the arcs are uncapacitated.

There are several variations of FCNDP in the literature, each of which involves a particular objective function and, possibly, additional constraints. Among these variations, we can find the shortest path problem, minimum spanning tree problem, vehicle routing problem, traveling salesman problem, and Steiner problem in graph [4; 16; 106]. Numerous applications can be found for network design problems, for instance in transportation systems and telecommunication networks (see [15; 106]).

We are interested in a specific variant of the FCNDP, called fixed charge network design problem with shortest path constraints (FCNDP-SPC)[72; 94], which consists of adding multiple shortest path problems to the original problem.

The FCNDP-SPC involves two distinct agents acting simultaneously rather than sequentially when making decisions. On the upper level, the leader (first agent) is in charge of designing a transportation subnetwork (*i.e.*, choosing a subset of edges to be opened) in order to minimize the sum of fixed and variable costs. In response, on the lower level, the follower (second agent) must choose a set of shortest paths in the subnetwork designed in the upper level, through which the commodities will be sent.

The inclusion of the shortest path requirement makes the problem more difficult to solve exactly. There are few works done for solving exactly the general case of FCNDP-SPC [5; 72; 94; 107; 110], and most of the research is dedicated to

a particular application. In the literature, the FCNDP-SPC was investigated for the transportation of hazardous materials: the Hazmat transport network design problem (HTNDP) [21; 55; 56; 79; 84; 133]. In this application, a given set of hazardous materials is required to be transported from an origin to a destination over a road network. The problem consists of selecting road segments to be opened by the government (the leader) that aim, on the one hand, to minimize the total risk for the population. On the other, the leader assumes that the carriers (the follower) choose the shortest path in the resultant network. This is a particular case of FCNDP-SPC, where the fixed cost of each edge is equal to zero.

Kara and Verter [84] were the first to pose the problem as a bi-level program. Then, in [55; 56; 84; 110] the problem is transformed into a single-level mixed-integer programming and the researchers focus on exact methods. More recently, Gzara [79] studied a combinatorial bi-level formulation for the problem and proposed a cutting plane algorithm.

## 3.2 Mathematical models

We consider a transportation network, which can be modeled by an undirected graph $G = (V, E)$, where $V$ represents the set of facilities and $E$ represents the connections between them. The connection edges are uncapacitated and undirected. Furthermore, we consider a set $K$ of commodities to be transported over the network (these commodities may represent physical goods as raw material for industry or hazardous material). Each commodity $k \in K$, has a flow $\phi^k$ to be delivered through a shortest path between its origin $o(k)$ and its destination $d(k)$. Let us define the set of arcs $A = \{(i, j), (j, i) : \{i, j\} \in E\}$. A length $c_{ij}$ and variable costs $g_{ij}^k$, $k \in K$, are associated to each arc $a = (i, j) \in A$. Also, for each edge $e = \{i, j\} \in E$, a fixed cost $f_e$ is associated, and we assume that

$c_{ij} = c_{ji}$. The sets of all arcs leaving and arriving at node $i$ are denoted by $\delta^+(i)$ and $\delta^-(i)$, respectively.

The FCNDP-SPC amounts to design a subnetwork (i.e., select a set of edges in $E$ to be opened), and to find for each commodity $k \in K$ a shortest path in the resultant network such that the sum of the fixed and variable costs is minimized.

To formulate the FCNDP-SPC, two types of variables are defined. We use binary variables $y \in \{0,1\}^{|E|}$ for the network construction such that:

$$y_e = \begin{cases} 1, & \text{if the edge } e \text{ is chosen as a part of the subnetwork,} \\ 0, & \text{otherwise.} \end{cases}$$

Besides, we use variables $x \in \{0,1\}^{|A|\times K}$ where $x_{ij}^k$ denotes if commodity $k$ is sent ($x_{ij}^k = 1$) through the directed arc $a = (i,j) \in A$ or not ($x_{ij}^k = 0$). In the rest of this section, we will present five different formulations for the FCNDP-SPC problem. The first one (Subsection 3.2.1) is a bi-level integer programming model [84]. This formulation is then transformed into two one-level models using optimality conditions of the second level problem (Subsections 3.2.2 and 3.2.3). Two binary integer programming formulations (BILP) are also presented: the cycle (Subsection 3.2.4) and the proposed path (Subsection 3.2.5) based formulations.

### 3.2.1 Bi-level formulation

In the FCNDP-SPC, each commodity $k \in K$ has to be transported through the shortest path between its origin $o(k)$ and its destination $d(k)$, forcing the addition of shortest path constraints to the general problem. Besides selecting a subset of $E$ with the minimum sum of fixed and variable costs (leader problem), we also need to guarantee that the shortest path is used for each commodity $k \in K$ (follower problem).

The FCNDP-SPC can be modeled as a bi-level mixed-integer programming problem [84], as follows:

$$\min \quad \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{(i,j) \in A} \phi^k g_{ij}^k x_{ij}^k \tag{3.1a}$$

$$\text{s.t.} \quad y_e \in \{0,1\}, \quad \forall e \in E, \tag{3.1b}$$

$$\min \quad \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k \tag{3.1c}$$

$$\text{s.t.} \sum_{(i,j) \in \delta^+(i)} x_{ij}^k - \sum_{(i,j) \in \delta^-(i)} x_{ji}^k = b_i^k, \qquad \forall i \in V, \forall k \in K, \tag{3.1d}$$

$$x_{ij}^k + x_{ji}^k \leq y_e, \qquad \forall e = \{i,j\} \in E, \forall k \in K, \tag{3.1e}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall (i,j) \in A, \forall k \in K. \tag{3.1f}$$

where, for $i \in V$ and $k \in K$

$$b_i^k = \begin{cases} 1, & \text{if } \ i = o(k), \\ -1, & \text{if } \ i = d(k), \\ 0, & \text{otherwise.} \end{cases}$$

The objective functions of the first and the second levels are presented on (3.1a) and (3.1c). In (3.1d), we have the flow conservation constraints while constraints (3.1e) do not allow a flow to use arcs whose corresponding edges are closed. Finally, the constraints (3.1f) and (3.1b) require the variables $x_{ij}^k$ and $y_e$ to be binary. As constraints (3.1d) and (3.1e) are defined by a totally unimodular matrix, the integrality of $x$ can be replaced by a non-negativity constraint.

Notice that, solving the follower problem is equivalent to solving $|K|$ shortest path problems independently.

### 3.2.2 One-level formulation

The FCNDP-SPC can be formulated as a one-level integer programming problem through replacing the follower problem by optimality conditions [48; 84]. This can be done by applying the fundamental theorem of duality and complementarity slackness theorem [16], as follows:

$$\min \quad \sum_{e\in E} f_e y_e + \sum_{k\in K}\sum_{(i,j)\in A} \phi^k g_{ij}^k x_{ij}^k \tag{3.2a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in\delta^+(i)} x_{ij}^k - \sum_{(i,j)\in\delta^-(i)} x_{ji}^k = b_i^k \qquad \forall i \in V, \forall k \in K, \tag{3.2b}$$

$$x_{ij}^k + x_{ji}^k \leq y_e, \qquad \forall e = \{i,j\} \in E, \forall k \in K, \tag{3.2c}$$

$$\pi_i^k - \pi_j^k - \lambda_{e(a)}^k \leq c_a, \qquad \forall a = (i,j) \in A, \forall k \in K, \tag{3.2d}$$

$$(y_e - x_{ij}^k - x_{ji}^k)\lambda_e^k = 0, \qquad \forall e = \{i,j\} \in E, \forall k \in K, \tag{3.2e}$$

$$(c_a - \pi_i^k + \pi_j^k + \lambda_{e(a)}^k)x_{ij}^k = 0, \qquad \forall a = (i,j) \in A, \forall k \in K, \tag{3.2f}$$

$$\lambda_e^k \geq 0, \qquad \forall e = \{i,j\} \in E, \forall k \in K, \tag{3.2g}$$

$$\pi_i^k \in \mathbb{R}, \qquad \forall i \in V, \forall k \in K, \tag{3.2h}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall (i,j) \in A, \forall k \in K, \tag{3.2i}$$

$$y_e \in \{0,1\}, \qquad \forall e \in E. \tag{3.2j}$$

Constraints (3.2b), (3.2c) and (3.2i) are the follower's constraints, and (3.2j) are the leader constraints. Considering an arc $a = (i,j) \in A$, we define the edge associated to $a$ by: $e(a) = \{i,j\}$. Constraints (3.2d)-(3.2f) represent the optimality conditions associated to the follower problem, which ensures the shortest path requirement.

This new formulation is no more linear since constraints (3.2e) and (3.2f) contain a product of variables. To bypass this problem, a big-M linearization is applied. After this modification, we can write the model as a one-level mixed-integer programming problem, as follows:

$$\min \quad \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{(i,j) \in A} \phi^k g_{ij}^k x_{ij}^k \tag{3.3a}$$

$$\text{s.t.} \quad (3.2b), (3.2c), (3.2d), (3.2j)$$

$$M y_e - M x_{ij}^k - M x_{ji}^k + \lambda_e^k \leq M, \qquad \forall e = \{i,j\} \in E, \forall k \in K, \tag{3.3b}$$

$$M x_{ij}^k - \pi_i^k + \pi_j^k + \lambda_e^k \leq M - c_a, \qquad \forall a = (i,j) \in A, \forall k \in K, \tag{3.3c}$$

$$\lambda_e^k \geq 0, \qquad \forall e = \{i,j\} \in E, \forall k \in K, \tag{3.3d}$$

$$\pi_i^k \in \mathbb{R}, \qquad \forall i \in V, \forall k \in K, \tag{3.3e}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall (i,j) \in A, \forall k \in K. \tag{3.3f}$$

Since the integrality of variables $x$ is assumed in the linearization, the constraints (3.3f) are added to the formulation. The parameter $M$ is a precomputed large number.

### 3.2.3 Bellman's Model

As we have mentioned before, optimality conditions for the lower level problem are, in fact, the optimality conditions of a set of shortest path problems. Hence, the FCNDP-SPC can be expressed in a more compact way [110], if we consider the Bellman's optimality conditions for the shortest path problem [4].

$$\min \quad \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{(i,j) \in A} \phi^k g_{ij}^k x_{ij}^k \tag{3.4a}$$

$$\text{s.t.} \quad (3.1d), (3.1e), (3.1b), (3.1f)$$

$$\pi_i^k - \pi_j^k \leq M - y_{e(a)}(M - c_a) - 2c_a x_{ji}^k, \qquad \forall a = (i,j) \in A, \forall k \in K, \tag{3.4b}$$

$$\pi_i^k \geq 0, \qquad \forall i \in V, \forall k \in K, \tag{3.4c}$$

$$\pi_{d(k)}^k = 0, \qquad \forall k \in K. \tag{3.4d}$$

Non-negative variables $\pi_i^k$ represent the shortest path distance between the node $i$ and $d(k)$ for each commodity $k$ (See, Section 2.3). Then, $\pi_{d(k)}^k$, $k \in K$, are set to be equal to zero in Constraints (5.6e). Constraints (3.4b) are the lifted version of Belman's optimality conditions that guarantee the shortest path requirement. As in the previous formulation, the parameter $M$ is a precomputed large value. To improve the quality of the formulation, we want to define the smallest possible value of $M$ (in order to strengthen the associated constraint). Let us take the constraint (3.4b), $M$ can take each value with:

$$\pi_i^k - \pi_j^k \leq M, \qquad \forall a = (i,j) \in A, \quad \forall k \in K.$$

A bound that can be used as a value for $M$ is:

$$M = \sum_{i \in V} \left( \max_{j:(i,j) \in A} c_{ij} \right) - \min_{(i,j) \in A} c_{ij}.$$

This value of $M$ is used in the implementation of all formulations in this paper including big-$M$ constraints.

### 3.2.4 BILP formulation based on cycle constraints

A one-level cycle based BILP formulation is proposed in [79] for the FCNDP-SPC. Before presenting this formulation, we first introduce some notations. Let $\Phi$ be the set of pairs $(x, y)$ satisfying the constraints of the first and the second level problems of the bi-level formulation:

$$\Phi = \{(x, y) : (3.1b), (3.1d), (3.1e), (3.1f)\}.$$

Let $\bar{y}$ be a decision of the leader and define the restricted graph $G(\bar{y}) = (V, E(\bar{y}))$ with $E(\bar{y}) = \{e \in E : \bar{y}_e = 1\}$. The feasible region of the follower,

denoted by $\Phi(\bar{y})$, is given by the set of all paths on $G(\bar{y})$, i.e.,

$$\Phi(\bar{y}) = \{x : (3.1\text{d}), (3.1\text{f}), x_{ij}^k + x_{ji}^k \leq \bar{y}_e, \forall e = \{i,j\} \in E, \forall k \in K\}.$$

The followers' reaction set $\Omega(\bar{y})$ is defined as the set of shortest paths, for all commodities in $K$, when the leader decision is $\bar{y}$:

$$\Omega(\bar{y}) = \arg \min_{x \in \Phi(\bar{y})} \{\sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ij}^k\}.$$

We consider the fixed charge network design problem (FCNDP) defined as:

$$\begin{aligned} \min \quad & \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{(i,j) \in A} \phi^k g_{ij}^k x_{ij}^k \\ \text{s.t.} \quad & (x, y) \in \Phi. \end{aligned}$$

Given a feasible solution $(\bar{x}, \bar{y}) \in \Phi$ of FCNDP, if $\bar{x} \notin \Omega(\bar{y})$ then, clearly, it is not feasible for the FCNDP-SPC. As a consequence, there exists at least one commodity $k \in K$ with alternative paths $p$ and $p'$ from $o(k)$ to $d(k)$ in the restricted graph $G(\bar{y})$ and these alternative paths have unequal costs (wrt the lengths $c_{ij}$). The alternative paths $p$ and $p'$ form at least one cycle defined by sub-paths $\tilde{p} \subseteq p$ and $\tilde{p}' \subseteq p'$ such that $c(p) > c(p')$.

In this case, the commodity $k$ will use the cheapest sub-path $p'$. The example in Figure 3.1 presents the case when a commodity with $o = 1$ and $d = 5$ has two paths in the restricted graph with different costs. The two paths form the cycle $(2 - 3 - 5 - 4 - 2)$. In this example, the commodity takes the shortest sub-path $p'$.

Let $P(k)$ denote the set of all paths in the original graph $G = (V, E)$ for the

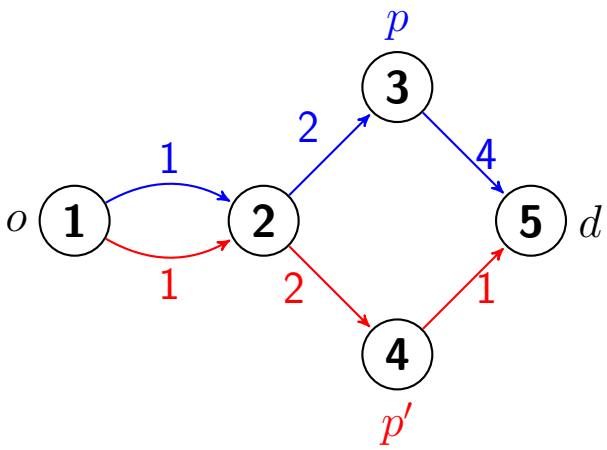



Figure 3.1: A restricted graph containing two alternative paths for a commodity to be sent from 1 to 5.

commodity $k$, i.e. all paths from $o(k)$ to $d(k)$. Let $|p|$ represent the number of arcs in a given path $p$. Using additional binary variables $z_p^k$, for each $k \in K$ and for each $p \in P(k)$, the FCNDP-SPC is formulated in [79] as follows.

$$\min \quad \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{(i,j) \in A} \phi^k g_{ij}^k x_{ij}^k \tag{3.5a}$$

$$\text{s.t.} \quad (3.1b), (3.1d), (3.1e), (3.1f) \tag{3.5b}$$

$$\sum_{(i,j) \in p} x_{ij}^k \leq |p| - 1 + z_p^k, \qquad \forall k \in K, \forall p \in P(k), \tag{3.5c}$$

$$z_p^k \leq x_{ij}^k, \qquad \forall k \in K, \forall p \in P(k), \forall (i,j) \in p, \tag{3.5d}$$

$$\sum_{e \in p'} y_e \leq |p'| - z_p^k, \qquad \forall k \in K, \forall p, p' \in P(k), \text{s.t. } c(p') < c(p), \tag{3.5e}$$

$$z_p^k \in \{0,1\}, \qquad \forall k \in K, \forall p \in P(k). \tag{3.5f}$$

The binary variable $z_p^k$ is equal to 1 if $p$ is the path used by commodity $k$, and 0 otherwise. Hence, a constraint in (3.5c) forces $z_p^k$ to take value 1 if $\sum_{(i,j) \in p} x_{ij}^k = |p|$. Likewise, a constraint in (3.5d) imposes $z_p^k = 0$ whenever there exists an arc

$(i,j) \in p$ such that $x_{ij}^k = 0$. Then, constraints (3.5e) eliminate any solution $(\bar{x}, \bar{y})$ that has alternative sub-paths with unequal costs. Notice that this formulation has a number of constraints and variables which depend on the number of paths for each commodity $k \in K$.

### 3.2.5 BILP formulation based on path constraints

In this subsection, we propose an alternative path based formulation to avoid the additional variables $z_p^k$, for each $k \in K$ and each $p \in P(k)$. We replace the set of constraints (3.5c)-(3.5e) by one set of constraints in charge of avoiding the commodities to use any path $p$ whenever a path $p'$ with $c(p') < c(p)$ is opened by the leader. The FCNDP-SPC is modeled as follows.

$$\min \quad \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{(i,j) \in A} \phi^k g_{ij}^k x_{ij}^k \tag{3.6a}$$

$$\text{s.t.} \quad (3.1\text{d}), (3.1\text{e})$$

$$\sum_{(i,j) \in A} c_{ij} x_{ij}^k \leq c(p) + (|p| - \sum_{e \in p} y_e) M, \qquad \forall k \in K, \forall p \in P(k), \tag{3.6b}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall (i,j) \in A, \forall k \in K, \tag{3.6c}$$

$$y_e \in \{0,1\}, \qquad \forall e \in E. \tag{3.6d}$$

Parameter $M$ is a precomputed large value (see Subsection 3.2.3). The above formulation contains a polynomial number of variables but a number of constraints that depends on the number of paths for each commodity $k \in K$.

## 3.3 Theoretical comparison and improvements

This section is devoted to compare the formulations (3.5a)-(3.5f) and (3.6a)-(3.6d) of FCNDP-SPC. As shown in the previous section, we have presented a cycle-based and a path-based formulations for the FCNDP-SPC. To compare the two

BILP formulations, we will study the relation between their sets of feasible points. First, let us rewrite the set of inequalities (3.5c)-(3.5e) as a single inequality that does not involve the variable $z_p^k$.

**Theorem 10.** *For each $p, p' \in P(k)$, such that $o(p) = o(p')$, $d(p) = d(p')$ and $c(p') < c(p)$, we can rewrite* (3.5c)-(3.5e) *as one set of inequalities:*

$$\sum_{(i,j)\in p} x_{ij}^k - |p| + 1 \leq |p'| - \sum_{e\in p'} y_e, \qquad \forall p, p' \in P(k), \forall k \in K. \tag{3.7}$$

*Proof.* Let $(P1)$ and $(P2)$ be the sets of points defined respectively by (3.5c)-(3.5e) and (3.7) associated with $p, p' \in P(k)$, i.e.,

$$(P1)\begin{cases} \sum\limits_{(i,j)\in p} x_{ij}^k \leq |p| - 1 + z_p^k, & \\ z_p^k \leq x_{ij}^k, & \forall (i,j) \in A, \forall k \in K, \\ \sum\limits_{e\in p'} y_e \leq |p'| - z_p^k, & \\ 0 \leq x_{ij}^k \leq 1, & \forall (i,j) \in A, \forall k \in K, \\ 0 \leq y_e \leq 1, & \forall e \in E, \forall k \in K, \\ 0 \leq z_p^k \leq 1. & \end{cases}$$

and

$$(P2)\begin{cases} \sum\limits_{(i,j)\in p} x_{ij}^k - |p| + 1 \leq |p'| - \sum\limits_{e\in p'} y_e, & \\ 0 \leq x_{ij}^k \leq 1, & \forall (i,j) \in A, \forall k \in K, \\ 0 \leq y_e \leq 1, & \forall e \in E. \end{cases}$$

The inequalities are equivalent if and only if $(P1) = (P2)$. By using the Fourier-Motzkin elimination method, we can eliminate variables $z_p^k$ from $(P1)$, and show

that $(P1)$ is equivalent to the following system of inequalities:

$$\sum_{(i,j)\in p} x_{ij}^k - |p| + 1 \leq |p'| - \sum_{(i,j)\in p'} y_{ij}, \tag{3.8a}$$

$$\sum_{(i,j)\in p} x_{ij}^k - |p| + 1 \leq x_{ij}^k, \qquad \forall (i,j) \in p, \tag{3.8b}$$

$$0 \leq x_{ij}^k \leq 1, \qquad \forall (i,j) \in A, \forall k \in K, \tag{3.8c}$$

$$0 \leq y_e \leq 1, \qquad \forall e \in E. \tag{3.8d}$$

We can easily check that the inequality (3.8b) is trivial. Therefore, the set of constraints of $(P1)$ is equivalent to the set of points defining $(P2)$ and $(P2) = (P1)$. □

Based on the result of Theorem 10, we investigate if a relation between the polytopes of the linear relaxation of formulations (3.5) and (3.6) can be established.

**Theorem 11.** *Let us define:*

$$\mathcal{P}1 = \begin{cases} (x_{ij}^k y_e) \in [0,1], \\ x \in \Phi(y), \\ \sum\limits_{(i,j)\in p} x_{ij}^k - |p| + 1 \leq |p'| - \sum\limits_{e\in p'} y_e, \qquad \forall p, p' \in P(k). \end{cases}$$

*and*

$$\mathcal{P}2 = \begin{cases} (x_{ij}^k, y_e) \in [0,1], \\ x \in \Phi(y), \\ \sum\limits_{(i,j)\in E} c_{ij} x_{ij}^k \leq C(p') + (|p'| - \sum\limits_{(i,j)\in p'} y_{ij})M, \qquad \forall p' \in P(k). \end{cases}$$

*then we have* $\mathcal{P}1 \not\subseteq \mathcal{P}2$ *and* $\mathcal{P}2 \not\subseteq \mathcal{P}1$.

*Proof.* We define an instance of the FCNDP-SPC problem with one commodity that is sent from an origin $s \in V$ to a destination node $t \in V$ via two alternative paths $p$ and $p'$ which form a cycle.

To show $\mathcal{P}1 \not\subseteq \mathcal{P}2$ (resp. $\mathcal{P}2 \not\subseteq \mathcal{P}1$), it is sufficient to find a fractional vector which is included in $\mathcal{P}1 \backslash \mathcal{P}2$ (resp. $\mathcal{P}2 \backslash \mathcal{P}1$).

- We define the fractional vector:

$$\begin{aligned} y_e^* &= 1 - \epsilon && \forall e \in E \\ x_{ij}^* &= 1 - \epsilon && \forall (i,j) \in p \\ x_{ij}^* &= \epsilon && \forall (i,j) \in p' \\ |p| &= 2 \, |p'| = 2 \\ c(p') &= c' \leq c = c(p) \end{aligned}$$

  The path constraint (3.6b) for this vector is:

$$(1-\epsilon)c + \epsilon c' \leq c' + \epsilon M$$

  Hence, the constraint is satisfied for many values of $\epsilon, M$ (we can take for instance: $\epsilon = \frac{1}{4}$ and $M = 4c$). However, the cycle constraint in $\mathcal{P}1$ for this vector becomes:

$$\frac{10}{4} \leq 2$$

  then, $(y*, x*) \notin \mathcal{P}1$.

  Hence, $(x^*, y^*) \in \mathcal{P}2 \backslash \mathcal{P}1$.

- We define the fractional vector:

$$\begin{aligned} y_e^* &= 1 - \epsilon && \forall e \in E \\ x_{ij}^* &= 1 - \epsilon && \forall (i,j) \in p \\ x_{ij}^* &= \epsilon && \forall (i,j) \in p' \\ |p'| &= 1 \\ c(p') &= c' = 1 \\ M &= c(p) = c = 4 \end{aligned}$$

We can easily check that $(x^*, y^*) \in \mathcal{P}1$, for all $\epsilon > \frac{1}{1+|p|}$. Furthermore, choosing $\epsilon = \frac{2}{5}$ and $|p| = 4$, the path constraint cannot be satisfied.

Hence, we obtain: $(x^*, y^*) \in \mathcal{P}1 \backslash \mathcal{P}2$.

□

Theorem 11 proves that the two BILP formulations (3.5) and (3.6) are not comparable.

In Theorem 10, we proved that the inequalities (3.7) can eliminate any path violating the shortest path requirement. Thus, the FCNDP-SPC can be formulated as a new cycle-based formulation:

$$\min \quad \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{(i,j) \in A} \phi^k g_{ij}^k x_{ij}^k, \tag{3.9a}$$

$$\text{s.t.} \quad \sum_{(i,j) \in \delta^+(i)} x_{ij}^k - \sum_{(i,j) \in \delta^-(i)} x_{ji}^k = b_i^k, \qquad \forall i \in V, \forall k \in K, \tag{3.9b}$$

$$x_{ij}^k + x_{ji}^k \leq y_e, \qquad \forall e \in E, \forall k \in K, \tag{3.9c}$$

$$\sum_{(i,j) \in p} x_{ij}^k + \sum_{e \in p'} y_e \leq |p| + |p'| - 1, \quad \forall k \in K, \forall p, p' \in P(k) : c(p') < c(p), \tag{3.9d}$$

$$x_{ij}^k \in \{0, 1\}, \qquad \forall (i,j) \in A, \forall k \in K, \tag{3.9e}$$

$$y_e \in \{0, 1\}, \qquad \forall e \in E. \tag{3.9f}$$

## 3.4 Proposed algorithms for FCNDP-SPC

In Sections 3.2 and 3.3, we presented five different formulations to this problem. Table 3.1 compares them according to the number of variables and constraints.

Formulations (4.3) and (3.4) are compact ones. On the other hand, formulations (3.5),(3.6), and (3.9) may have exponential numbers of constraints. Formulation (3.5) may also have an exponential number of variables.

| Formulations | Number of binary variables | Number of constraints |
|---|---|---|
| One-level formulation | $K\|A\| + \|E\|$ | $K(\|V\| + 2\|A\| + 2\|E\|)$ |
| Bellman formulation | $K\|A\| + \|E\|$ | $K(\|V\| + \|A\| + \|E\|)$ |
| Cycle-based formulation | $K(\|A\| + \|P(k)\|) + \|E\|$ | $K(\|V\| + \|E\| + 3\|P(k)\|)$ |
| Path-based formulation | $K\|A\| + \|E\|$ | $K(\|V\| + \|E\| + \|P(k)\|)$ |
| New cycle-based formulation | $K\|A\| + \|E\|$ | $K(\|V\| + \|E\| + \|P(k)\|)$ |

Table 3.1: Number of constraints and binary variables of each formulation

In this section, we focus on presenting different exact methods to solve the FCNDP-SPC.

### 3.4.1 Compact formulations

The first way to solve the FCNDP-SPC is to feed the models (4.3) and (3.4) to Gurobi solver with default parameters, resulting into the approaches denoted **B&B1** and **B&B2** denote the branch-and-bound algorithms for the formulations (4.3) and (3.4), respectively.

### 3.4.2 Iterative cutting plane algorithms

Our iterative cutting plane algorithms are based on formulations (3.5),(3.6) and (3.9). Algorithm 7 gives the pseudo-code of these algorithms.

**Algorithm 7** Cutting plane algorithms

**Step 0:** $(BILP)^0$ is the fixed charge network design problem defined by the set of constraints (3.1b),(3.1d),(3.1e), and (3.1f).

$l = 0$.

**Step 1:** Solve $(BILP)^l$ to obtain the solution $(x^l, y^l)$. Let $N^l$ be the network induced by $y^l$, and $(P^k)^l$ be the obtained path for the commodity $k \in K$. We denote by $c^l(P^k)$ the total cost of $(P^k)^l$.

**Step 2:** On $N^l$, find $(P'^k)^l$ the shortest path for each commodity $k$ with cost $c^l(P'^k)$.

**Step 3:** If $c^l(P^k) = c^l(P'^k)$ for each commodity $k$, then $N^l$ defines an optimal solution for FCNDP-SPC, Stop.

**Step 4:** For the commodities with $c^l(P^k) \neq c^l(P'^k)$, generate a set of shortest path constraints to eliminate the path $(P^k)^l$. Let $S$ be the set of constraints generated. Append the constraints generated $S$ to $(BILP)^{l+1}$.

Do $l = l + 1$, and go to step 1.

The set $S$ of inequalities in **Step 4** can be generated in three different ways giving us three different iterative cutting plane methods **CP1**, **CP2**, and **CP3**:

**CP1**: Generate the inequality (3.6b).

**CP2**: Find the cycles formed by $(P'^k)^l$ and $(P^k)^l$. Generate the set of valid inequalities (3.5c)-(3.5e).

**CP3**: Find the cycles formed by $(P'^k)^l$ and $(P^k)^l$. Generate the inequality (3.7).

### 3.4.3 Branch-and-cut algorithm

The FCNDP-SPC can also be solved to optimality by using branch-and-cut algorithms based on formulations (3.6) and (3.9).

The principle of the algorithm used here is to solve a fixed charge network design problem (without the shortest path constraints) by a branch-and-bound procedure and to add valid inequalities to each integral node violating the shortest path constraints. The algorithm stops when there is no more node to evaluate, i.e., all the need path constraints were generated. The difference of this algorithm over the iterative cutting plane algorithms lies in solving a unique mixed-integer problem by considering all the shortest path constraints. Next, we detail each component of the Branch-and-cut procedure.

**The initial model:** The initial model consists in minimizing the sum of the fixed and variable costs under the flow conservation constraints (3.1d), the constraints (3.1e) forcing the flow to use only the opened edges and the binary requirement constraints of $x$ (3.1f) and of $y$ (3.1b).

**Separation problem:** Since the initial problem does not contain all the constraints of the FCNDP-SPC, an integer solution obtained can be infeasible. For this reason, for each integer node on the branch-and-bound tree, we introduce a cut generation procedure to eliminate the infeasible paths.

We first check if an integer solution $(\bar{x}, \bar{y})$ is feasible for FCNDP-SPC by solving a set of shortest path problems. For each commodity $k$, in the subnetwork defined by $\bar{y}$ we verify if the path $P'$ defined by $\bar{x}$ is the shortest path from $o(k)$ to $d(k)$ in $\bar{y}$. If $(\bar{x}, \bar{y})$ is infeasible, i.e., there exists at least one path $P$ with $c(P) < c(P')$, then a set of shortest path constraints is added. Two variants of the algorithm were developed. In **B&C1** shortest path constraints correspond to (3.6b) while for **B&C2** they correspond to (3.7).

### 3.4.4 Valid inequalities

The different formulations of the FCNDP-SPC can be weak. One way to strengthen the models is to introduce a set of valid inequalities. From the definition of the

problem, we can remark that, if there exist several different commodities with the same origin and destination, then their paths in the optimal solution have the same cost.

The following proposition shows that a valid inequality can be generated in this particular case.

**Proposition 1.** *Consider two commodities $k^1, k^2$ such that $o(k^1) = o(k^2)$ and $d(k^1) = d(k^2)$, then the constraint:*

$$\sum_{(i,j)\in A} c_{ij} x_{ij}^{k^1} = \sum_{(i,j)\in A} c_{ij} x_{ij}^{k^2} \tag{3.10}$$

*is valid for $FCNDP - SPC$.*

*Proof.* Straightforward. □

## 3.5 Numerical results

In this section, we present computational experiments carried out with all the methods described in the previous sections. All algorithms are implemented in Julia 0.5.0, and the problems are solved using Gurobi 6.5.2 (with four threads). Simulations were performed on an Intel(R)core TM i7-3520M CPU@2.90 GHz×4 computer with 8 GB of RAM. Numerical experiments were performed on two sets of data. The first one concerns 405 random instances generated for this work, while the second one consists of real instances from different city transportation networks (Ravenna, Italy [23], and Albany, NY, USA [131]). Next, we describe each data set and discuss the computational results obtained on each one.

We summarize all the exact algorithms used to solve the FCNDP-SPC:

**B&B1:** Compact formulation (4.3).

**B&B2:** Compact formulation (3.4).

**CP1:** Cutting plane algorithm using the inequalities (3.6b).

**CP2:** Cutting plane algorithm using the inequalities (3.5c)-(3.5e).

**CP3:** Cutting plane algorithm using the inequalities (3.7).

**B&C1:** Branch-and-cut algorithm using inequalities (3.6b) in the cut generation.

**B&C2:** Branch-and-cut algorithm using inequalities (3.7) in the cut generation.

### 3.5.1 Random instances

The different methods are tested on 405 instances generated randomly with different values for the angles $\alpha$ between the variable cost vector "$g$" and the length vector associated to the edges "$c$" in the network $G(V, E)$. We consider three different categories for the value of $\alpha$:

- $0° \leq \alpha \leq 10°$
- $40° \leq \alpha \leq 50°$
- $80° \leq \alpha \leq 90°$

The purpose of distinguishing these three scenarios is to examine whether the difficulty of the instance is related to the angle between $g$ and $c$, i.e., the direction of the objective function of the upper and the lower levels. For instance, we can expect that if both levels go in the same direction, i.e., the two vectors are very close to each other, the instance is easy.

The instances are generated varying also the number of nodes in the graph $n \in \{10, 20, 30\}$, the graph density $d \in \{0.3, 0.5, 0.7\}$, and the number of different

Table 3.2: CPU time in seconds for instances with $0° \leq \alpha \leq 10°$

| $n$ - $d$ - $K$ | B&B1 | B&B2 | CP1 | CP2 | CP3 | B&C1 | B&C2 |
|---|---|---|---|---|---|---|---|
| 10-0.3-5 | 0.04 | 0.01 | 0.01 | 0.01 | 0.01 | 0.08 | 0.13 |
| 10-0.3-10 | 0.09 | 0.04 | 0.02 | 0.02 | 0.02 | 0.06 | 0.13 |
| 10-0.3-15 | 0.26 | 0.06 | 0.05 | 0.07 | 0.05 | 0.20 | 0.28 |
| 10-0.5-5 | 0.07 | 0.03 | 0.03 | 0.05 | 0.04 | 0.04 | 0.09 |
| 10-0.5-10 | 0.43 | 0.12 | 0.14 | 0.25 | 0.19 | 0.22 | 0.35 |
| 10-0.5-15 | 0.82 | 0.45 | 0.19 | 0.30 | 0.21 | 0.45 | 0.94 |
| 10-0.7-5 | 0.04 | 0.02 | 0.01 | 0.01 | 0.01 | 0.05 | 0.05 |
| 10-0.7-10 | 0.47 | 0.25 | 0.08 | 0.20 | 0.14 | 0.31 | 0.40 |
| 10-0.7-15 | 0.63 | 0.20 | 0.13 | 0.16 | 0.15 | 0.30 | 0.49 |
| 10-Average | 0.32(45) | 0.13(45) | 0.07(45) | 0.12(45) | 0.09(45) | 0.19(45) | 0.32(45) |
| 20-0.3-10 | 0.86 | 0.21 | 0.10 | 0.09 | 0.07 | 0.41 | 0.47 |
| 20-0.3-20 | 284.73 | 8.80 | 4.13 | 12.15 | 28.61 | 3.07 | 10.27 |
| 20-0.3-30 | 655.37 | 61.48 | 9.07 | 54.10 | 45.68 | 8.99 | 34.63 |
| 20-0.5-10 | 1.49 | 0.19 | 0.17 | 0.19 | 0.14 | 0.40 | 0.48 |
| 20-0.5-20 | 257.22 | 10.82 | 4.93 | 14.70 | 11.75 | 5.20 | 7.13 |
| 20-0.5-30 | 379.72 (2) | 1095.67(4) | 349.24 | 455.11 | 453.08 | 440.30 | 633.17(4) |
| 20-0.7-10 | 0.93 | 0.20 | 0.12 | 0.14 | 0.11 | 0.33 | 0.40 |
| 20-0.7-20 | 308.80 | 56.19 | 23.96 | 80.47 | 65.89 | 13.70 | 23.79 |
| 20-0.7-30 | 622.18(1) | 912.28 | 241.56 | 547.02 | 542.20 | 139.11 | 694.23 |
| 20-Average | 279.03(38) | 238.43(44) | 70.36(45) | 129.33(45) | 127.50(45) | 67.95(45) | 156.06(44) |
| 30-0.3-15 | 38.26 | 1.07 | 0.49 | 0.66 | 0.65 | 1.31 | 1.32 |
| 30-0.3-30 | 1595.92(2) | 273.98 | 65.70 | 179.70 | 152.62 | 54.21 | 67.33 |
| 30-0.5-15 | 22.78 | 1.18 | 0.55 | 0.65 | 0.64 | 2.32 | 1.55 |
| 30-0.5-30 | - | - | 862.61 | 761.85 | 729.97 | - | - |
| 30-0.7-15 | 132.37 | 5.65 | 4.26 | 4.77 | 4.39 | 8.22 | 3.64 |
| 30-0.7-30 | - | - | 1095.17 | 1074.14 | 986.96 | - | - |
| 30-Average | 447.33(17) | 70.47(20) | 338.13(30) | 336.96(30) | 312.54(30) | 16.51(20) | 18.46(20) |

commodities to be transported $K \in \{\frac{n}{2}, n, \frac{3n}{2}\}$. For each combination of $(n, d, K)$, 5 instances are generated.

Similar random instances have also been generated for FCNDP-SPC by Mauttone et al [110] and used in [72]. Their instances were not used here for two reasons. First, these instances belong to a particular case of the problem where all the variable costs are equal for all the commodities. Second, all instances are included in the first scenario of $\alpha$, where $\alpha$ is close to zero. Hence, the random instances of [110] are considered very easy to solve.

The obtained results are shown in Tables 3.2, 3.3, and 3.4, where, we report in each row the average CPU time in seconds spent by each algorithm on each group

Table 3.3: CPU time in seconds for instances with $40° \leq \alpha \leq 50°$

| $n$ - $d$ - $K$ | B&B1 | B&B2 | CP1 | CP2 | CP3 | B&C1 | B&C2 |
|---|---|---|---|---|---|---|---|
| 10-0.3-5 | 0.04 | 0.02 | 0.01 | 0.01 | 0.01 | 0.04 | 0.07 |
| 10-0.3-10 | 0.16 | 0.05 | 0.05 | 0.05 | 0.04 | 0.17 | 0.19 |
| 10-0.3-15 | 0.28 | 0.09 | 0.10 | 0.14 | 0.10 | 0.30 | 0.44 |
| 10-0.5-5 | 0.02 | 0.08 | 0.04 | 0.02 | 0.01 | 0.05 | 0.08 |
| 10-0.5-10 | 0.24 | 0.44 | 0.35 | 0.39 | 0.19 | 0.31 | 0.47 |
| 10-0.5-15 | 0.77 | 6.60 | 1.34 | 1.91 | 1.01 | 0.90 | 1.11 |
| 10-0.7-5 | 0.02 | 0.04 | 0.01 | 0.02 | 0.01 | 0.06 | 0.06 |
| 10-0.7-10 | 0.47 | 7.86 | 1.17 | 3.51 | 1.43 | 0.86 | 0.92 |
| 10-0.7-15 | 4.30 | 256.09 | 9.03 | 29.70 | 11.85 | 3.87 | 12.62 |
| 10-Average | 0.70(45) | 30.14(45) | 1.35(45) | 3.97(45) | 1.63(45) | 0.73(45) | 1.77(45) |
| 20-0.3-10 | 1.65 | 0.25 | 0.16 | 0.20 | 0.15 | 0.53 | 0.52 |
| 20-0.3-20 | 1226.54 | 27.66 | 43.71 | 117.58 | 94.21 | 16.77 | 35.32 |
| 20-0.3-30 | - | 1427.53(4) | 827.44 | 696.42 | 648.77 | 927.19 | 1147.03(3) |
| 20-0.5-10 | 0.26 | 4.09 | 0.82 | 0.28 | 0.20 | 1.00 | 0.51 |
| 20-0.5-20 | 271.99(4) | 308.38(2) | 659.96(4) | 329.23(4) | 132.99 | 1182.37 | 841.59 |
| 20-0.5-30 | - | 3410.90 | 614.23 | 500.25 | 776.10 | - | - |
| 20-0.7-10 | 0.42 | 2.74 | 0.91 | 0.28 | 0.19 | 0.73 | 0.58 |
| 20-0.7-20 | 185.35(4) | 648.43(2) | 334.97 | 224.35 | 210.19 | 881.23(4) | 386.98(4) |
| 20-0.7-30 | - | - | 1196.37 | 1030.52 | 910.30 | - | - |
| 20-Average | 281.04(28) | 728.75(33) | 408.73(44) | 322.12(44) | 308.12(45) | 429.97(34) | 344.65(32) |
| 30-0.3-15 | 19.79 | 856.09 | 4.08 | 6.55 | 6.26 | 15.32 | 8.17 |
| 30-0.3-30 | - | - | - | - | 744.85 | - | - |
| 30-0.5-15 | 4.07 | 192.25 | 1.54 | 1.45 | 1.47 | 5.95 | 2.55 |
| 30-0.5-30 | - | - | - | - | 785.04 | - | - |
| 30-0.7-15 | 27.63 | 1171.09 | 5.82 | 12.69 | 11.34 | 24.51 | 12.80 |
| 30-0.7-30 | - | - | - | - | 674.06 | - | - |
| 30-Average | 17.16(15) | 739.81(15) | 3.81(15) | 6.90(15) | 370.50(30) | 15.26(15) | 7.84(15) |

of 5 instances with the same $(n, d, k)$. The symbol "-" means that the algorithm was not able to find the optimal solution in the time limit of $3600s$ for all the considered instances, also, an additional number (.) is added to represent the number of instances solved in each group of 5 instances. In addition, the average CPU time and the total number of solved instances are given for each number of nodes $n$. Overall, the iterative cutting plane algorithms **CP1** and **CP3** outperform the other algorithms and they solve more instances. Furthermore, algorithms **B&C1** and **B&C2** are more efficient than **B&B1** and **B&B2**, although the difference is less marked. We can also see from Tables 3.2, 3.3, and 3.4 that the instances with $0° \leq \alpha \leq 10°$ are easy to solve than the other instances.

Table 3.4: CPU time in seconds for instances with $80° \leq \alpha \leq 90°$

| $n$ - $d$ - $K$ | B&B1 | B&B2 | CP1 | CP2 | CP3 | B&C1 | B&C2 |
|---|---|---|---|---|---|---|---|
| 10-0.3-5 | 0.03 | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 | 0.06 |
| 10-0.3-10 | 0.17 | 0.05 | 0.06 | 0.05 | 0.03 | 0.14 | 0.22 |
| 10-0.3-15 | 0.40 | 0.12 | 0.11 | 0.10 | 0.07 | 0.26 | 0.36 |
| 10-0.5-5 | 0.02 | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 | 0.05 |
| 10-0.5-10 | 0.46 | 0.12 | 0.14 | 0.27 | 0.18 | 0.27 | 0.29 |
| 10-0.5-15 | 3.18 | 0.37 | 0.85 | 1.46 | 1.08 | 0.52 | 1.29 |
| 10-0.7-5 | 0.07 | 0.02 | 0.02 | 0.03 | 0.02 | 0.05 | 0.73 |
| 10-0.7-10 | 0.87 | 0.16 | 0.22 | 0.35 | 0.22 | 0.30 | 3.61 |
| 10-0.7-15 | 108.61 | 1.96 | 2.11 | 9.41 | 5.38 | 0.80 | 89.82 |
| 10-Average | 12.65(45) | 0.31(45) | 0.39(45) | 1.30(45) | 0.78(45) | 0.27(45) | 10.71(45) |
| 20-0.3-10 | 1.35 | 0.15 | 0.17 | 0.15 | 0.12 | 0.33 | 0.51 |
| 20-0.3-20 | 837.97(4) | 27.22 | 31.18 | 85.09 | 77.47 | 70.48 | 20.97 |
| 20-0.3-30 | - | 923.06 | 279.67 | 292.00 | 269.23 | 72.24(4) | 91.21 |
| 20-0.5-10 | 3.21 | 0.55 | 0.51 | 0.84 | 0.71 | 0.72 | 0.76 |
| 20-0.5-20 | 1161.59 | 41.18 | 22.61 | 50.50 | 39.08 | 18.35 | 33.82 |
| 20-0.5-30 | 33.02(4) | 264.56(2) | 943.67 | 655.29 | 644.81 | 1043.20(4) | 583.20(4) |
| 20-0.7-10 | 1.15 | 0.30 | 0.12 | 0.11 | 0.09 | 0.37 | 7.32 |
| 20-0.7-20 | 329.74(3) | 548.91 | 209.45 | 130.58 | 131.29 | 120.45 | 2.59(4) |
| 20-0.7-30 | - | - | 1444.21 | 885.10 | 893.73 | 1206.02(4) | 2508.14(4) |
| 20-Average | 338.29(31) | 225.74(37) | 325.73(45) | 233.30(45) | 228.50(45) | 281.89(42) | 360.94(42) |
| 30-0.3-15 | 87.28 | 4.07 | 1.79 | 2.46 | 2.19 | 4.43 | 4.99 |
| 30-0.3-30 | - | - | - | - | 771.18 | - | - |
| 30-0.5-15 | 159.66 | 1.59 | 1.16 | 1.07 | 1.05 | 2.41 | 1.51 |
| 30-0.5-30 | - | - | - | - | 737.51 | - | - |
| 30-0.7-15 | 9.62 | 1.00 | 0.50 | 0.51 | 0.46 | 1.39 | 1.10 |
| 30-0.7-30 | - | - | - | - | 688.11 | - | - |
| 30-Average | 85.52(15) | 2.22(15) | 1.15(15) | 1.35(15) | 366.75(30) | 2.74(15) | 2.53(15) |

We provide next more detailed statistics to compare the different formulations and algorithms. These are reported in Figure 3.2, while the full details are reported to Tables 3.5-3.8.

In order to compare the different one-level formulations presented in this work, in Figure 3.2a, we compare the average of the gap between the solution of the linear relaxation and the optimal solution. We can remark that the linear relaxations of the BILP formulations (3.5) and (3.9) are stronger than those of formulations (4.3) and (3.4). For instance, in the first case of $\alpha$ the gap of (3.5) (resp. (3.9)) is twice (resp. four times) smaller than the gap of the formulations (4.3) and (3.4). Although the formulation (3.9) has a smaller gap for our random

Table 3.5: Gap between the optimal solution and the linear relaxation

| Angle | $0° \leq \alpha \leq 10°$ | | | | $40° \leq \alpha \leq 50°$ | | | | $80° \leq \alpha \leq 90°$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Formulation | (4.3) | (3.4) | (3.5) | (3.9) | (4.3) | (3.4) | (3.5) | (3.9) | (4.3) | (3.4) | (3.5) | (3.9) |
| 10-3-5 | 0.00 | 0.00 | 0.00 | 0.00 | 1.66 | 1.60 | 0.43 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 10-3-10 | 0.86 | 0.87 | 0.54 | 0.00 | 1.88 | 1.83 | 0.00 | 0.00 | 3.42 | 3.41 | 2.13 | 0.00 |
| 10-3-15 | 1.91 | 1.96 | 0.17 | 0.00 | 5.78 | 5.68 | 1.02 | 0.00 | 4.22 | 4.20 | 1.16 | 0.00 |
| 10-5-5 | 2.68 | 2.68 | 2.52 | 0.00 | 0.75 | 0.75 | 0.76 | 0.76 | 0.00 | 0.00 | 0.00 | 0.00 |
| 10-0.5-10 | 2.39 | 2.40 | 1.28 | 0.90 | 6.26 | 6.27 | 6.30 | 6.30 | 5.52 | 5.41 | 1.65 | 0.96 |
| 10-0.5-15 | 4.68 | 4.70 | 1.14 | 0.42 | 7.99 | 8.00 | 8.01 | 8.01 | 10.43 | 10.37 | 6.07 | 3.66 |
| 10.7.5 | 0.42 | 0.42 | 0.00 | 0.00 | 0.46 | 0.46 | 0.00 | 0.00 | 1.57 | 1.57 | 0.68 | 0.00 |
| 10.7.10 | 8.15 | 8.16 | 2.54 | 2.42 | 9.46 | 9.46 | 4.77 | 3.50 | 4.50 | 4.38 | 1.81 | 1.34 |
| 10.7.15 | 2.68 | 2.68 | 0.56 | 0.00 | 16.88 | 16.89 | 11.78 | 9.29 | 12.33 | 12.44 | 5.98 | 5.10 |
| 20-0.3-10 | 0.83 | 0.83 | 0.24 | 0.00 | 1.16 | 1.15 | 0.37 | 0.00 | 2.38 | 2.38 | 0.73 | 0.00 |
| 20-0.3-20 | 7.54 | 7.54 | 5.17 | 3.19 | 10.06 | 10.04 | 7.60 | 6.46 | 9.17 | 9.15 | 7.71 | 6.57 |
| 20-0.3-30 | 6.96 | 6.97 | 4.17 | 2.64 | 13.67 | 13.65 | 11.46 | 10.60 | 8.97 | 8.97 | 7.19 | 6.31 |
| 20-0.5-10 | 0.50 | 0.50 | 0.28 | 0.00 | 2.31 | 2.31 | 2.33 | 2.33 | 3.66 | 3.65 | 1.69 | 1.22 |
| 20-0.5-20 | 6.44 | 6.44 | 3.44 | 2.02 | 11.94 | 11.99 | 12.04 | 12.04 | 5.56 | 5.57 | 4.23 | 3.02 |
| 20-0.5-30 | 0.08 | 0.08 | 0.06 | 0.06 | 11.86 | 11.83 | 11.89 | 11.89 | 10.28 | 10.29 | 8.36 | 6.79 |
| 20.7.10 | 0.73 | 0.73 | 0.00 | 0.00 | 2.73 | 2.74 | 1.35 | 0.00 | 1.02 | 1.02 | 0.00 | 0.00 |
| 20.7.20 | 6.49 | 6.51 | 4.29 | 3.72 | 8.98 | 8.99 | 7.21 | 6.13 | 6.31 | 6.32 | 4.41 | 3.40 |
| 20.7.30 | 0.10 | 0.10 | 0.07 | 0.06 | 9.97 | 9.98 | 8.00 | 7.22 | 8.82 | 8.82 | 7.45 | 6.22 |
| 30-0.3- 15 | 2.24 | 2.24 | 1.60 | 0.00 | 3.18 | 3.18 | 2.61 | 1.53 | 2.41 | 2.41 | 1.36 | 0.74 |
| 30-0.5-15 | 1.83 | 1.84 | 0.53 | 0.00 | 2.11 | 2.11 | 2.12 | 2.12 | 2.65 | 2.65 | 0.96 | 0.00 |
| 30.7.15 | 4.32 | 4.32 | 2.63 | 1.27 | 4.11 | 4.12 | 2.81 | 1.11 | 1.20 | 1.20 | 0.32 | 0.00 |
| Average | 2.94 | 2.95 | 1.49 | 0.79 | 6.34 | 6.33 | 4.90 | 4.25 | 4.97 | 4.96 | 3.04 | 2.16 |

instances, there are instances where the linear relaxation of the formulation (3.5) is stronger (see Theorem 11).

The computational experiments show that, overall, the iterative cutting plane algorithm **CP3** is faster than the branch-and-cut algorithms. One of the reasons that explain those results is the time consumed on the separation problem in each algorithm. To study the effect of the separation problem on the results, we compute the percentage of the run time consumed by the separation problem on the CPU-total in Figure 3.2b. This percentage is similar and negligible for the iterative cutting plane algorithms, unlike the branch-and-cut algorithms where the solution of separation problem takes for many instances more than 20% (see Figure 3.2d) of the total time consumed by the **B&C2** algorithm. To complement these results we compare the number of iterations and the number of cuts

Table 3.6: % of CPU-SP/CPU-total

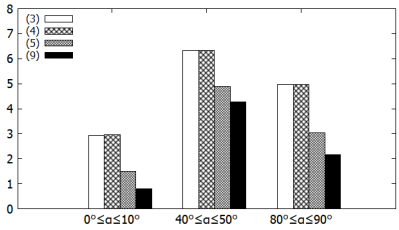
| Angle | $0° \leq \alpha \leq 10°$ | | | | $40° \leq \alpha \leq 50°$ | | | | $80° \leq \alpha \leq 90°$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ - $d$ - $K$ | CP1 | CP2 | CP3 | B&C1 | B&C2 | CP1 | CP2 | CP3 | B&C1 | B&C2 | CP1 | CP2 | CP3 | B&C1 | B&C2 |
| 10-0.3-5 | 0.28 | 0.27 | 0.27 | 0.00 | 0.01 | 0.24 | 0.25 | 0.22 | 0.01 | 0.01 | 0.28 | 0.26 | 0.26 | 0.00 | 0.01 |
| 10-0.3-10 | 0.30 | 0.31 | 0.31 | 0.01 | 0.02 | 0.25 | 0.28 | 0.29 | 0.03 | 0.03 | 0.27 | 0.28 | 0.30 | 0.02 | 0.03 |
| 10-0.3-15 | 0.32 | 0.32 | 0.30 | 0.03 | 0.04 | 0.25 | 0.25 | 0.27 | 0.05 | 0.07 | 0.23 | 0.28 | 0.29 | 0.04 | 0.05 |
| 10-0.5-5 | 0.22 | 0.22 | 0.23 | 0.01 | 0.01 | 0.45 | 0.31 | 0.26 | 0.01 | 0.01 | 0.29 | 0.28 | 0.27 | 0.00 | 0.01 |
| 10-0.5-10 | 0.23 | 0.24 | 0.25 | 0.04 | 0.05 | 0.29 | 0.38 | 0.21 | 0.04 | 0.06 | 0.18 | 0.22 | 0.23 | 0.04 | 0.04 |
| 10-0.5-15 | 0.19 | 0.22 | 0.24 | 0.06 | 0.16 | 0.26 | 0.19 | 0.13 | 0.12 | 0.13 | 0.09 | 0.08 | 0.08 | 0.09 | 0.19 |
| 10-0.7-5 | 0.26 | 0.27 | 0.27 | 0.01 | 0.01 | 0.41 | 0.26 | 0.28 | 0.01 | 0.01 | 0.23 | 0.22 | 0.19 | 0.01 | 0.01 |
| 10-0.7-10 | 0.20 | 0.20 | 0.19 | 0.04 | 0.06 | 0.20 | 0.26 | 0.14 | 0.09 | 0.10 | 0.16 | 0.17 | 0.19 | 0.04 | 0.08 |
| 10-0.7-15 | 0.25 | 0.24 | 0.23 | 0.04 | 0.07 | 0.04 | 0.02 | 0.02 | 0.28 | 0.53 | 0.06 | 0.05 | 0.06 | 0.12 | 0.45 |
| 20-0.3-10 | 0.19 | 0.21 | 0.22 | 0.04 | 0.04 | 0.18 | 0.19 | 0.21 | 0.07 | 0.05 | 0.17 | 0.18 | 0.20 | 0.04 | 0.05 |
| 20-0.3-20 | 0.07 | 0.09 | 0.07 | 0.40 | 1.27 | 0.02 | 0.03 | 0.03 | 0.82 | 3.00 | 0.01 | 0.01 | 0.01 | 0.85 | 1.80 |
| 20-0.3-30 | 0.03 | 0.03 | 0.03 | 0.73 | 4.06 | 0.00 | 0.00 | 0.00 | 2.86 | 32.67 | 0.00 | 0.00 | 0.00 | 1.36 | 3.82 |
| 20-0.5-10 | 0.16 | 0.16 | 0.18 | 0.04 | 0.05 | 0.20 | 0.26 | 0.17 | 0.09 | 0.06 | 0.10 | 0.11 | 0.12 | 0.07 | 0.08 |
| 20-0.5-20 | 0.06 | 0.07 | 0.08 | 0.36 | 0.41 | 0.01 | 0.01 | 0.01 | 1.78 | 8.97 | 0.03 | 0.03 | 0.04 | 0.46 | 2.42 |
| 20-0.5-30 | 0.01 | 0.01 | 0.01 | 1.21 | 7.91 | 0.00 | 0.00 | 0.00 | 2.01 | 12.47 | 0.00 | 0.00 | 0.00 | 1.71 | 21.05 |
| 20-0.7-10 | 0.14 | 0.15 | 0.16 | 0.03 | 0.04 | 0.11 | 0.25 | 0.14 | 0.05 | 0.07 | 0.11 | 0.12 | 0.13 | 0.04 | 0.04 |
| 20-0.7-20 | 0.04 | 0.05 | 0.05 | 0.51 | 1.59 | 0.02 | 0.04 | 0.02 | 2.94 | 15.14 | 0.02 | 0.03 | 0.03 | 0.49 | 2.28 |
| 20-0.7-30 | 0.00 | 0.00 | 0.00 | 1.58 | 5.53 | 0.00 | 0.00 | 0.00 | 4.16 | 28.09 | 0.00 | 0.00 | 0.00 | 1.70 | 15.99 |
| 30-0.3-15 | 0.12 | 0.12 | 0.12 | 0.14 | 0.15 | 0.05 | 0.05 | 0.05 | 0.81 | 0.64 | 0.09 | 0.10 | 0.11 | 0.34 | 0.58 |
| 30-0.3-30 | 0.02 | 0.02 | 0.02 | 1.91 | 3.93 | - | - | - | - | - | - | - | - | - | - |
| 30-0.5-15 | 0.09 | 0.11 | 0.11 | 0.26 | 0.23 | 0.07 | 0.08 | 0.08 | 0.28 | 0.13 | 0.08 | 0.09 | 0.09 | 0.23 | 0.21 |
| 30-0.5-30 | 0.00 | 0.00 | 0.00 | 2.88 | 24.01 | - | - | - | - | - | - | - | - | - | - |
| 30-0.7-15 | 0.04 | 0.04 | 0.05 | 0.44 | 0.33 | 0.03 | 0.03 | 0.04 | 0.69 | 0.75 | 0.09 | 0.11 | 0.10 | 0.08 | 0.08 |
| 30-0.7-30 | 0.00 | 0.00 | 0.00 | 2.41 | 23.24 | - | - | - | - | - | - | - | - | - | - |
| Average | 0.13 | 0.14 | 0.14 | 0.55 | 3.05 | 0.15 | 0.15 | 0.12 | 0.82 | 4.90 | 0.12 | 0.12 | 0.13 | 0.37 | 2.35 |

Table 3.7: Number of cuts generated by the cutting plane and the Branch-and-Cut algorithms

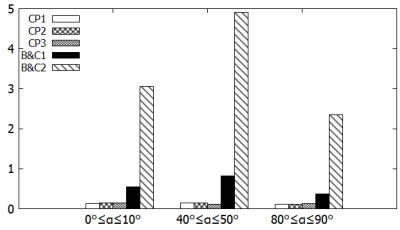
| Angle | $0° \leq \alpha \leq 10°$ | | | | | $40° \leq \alpha \leq 50°$ | | | | | $80° \leq \alpha \leq 90°$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ - $d$ - $K$ | CP1 | CP2 | CP3 | B&C1 | B&C2 | CP1 | CP2 | CP3 | B&C1 | B&C2 | CP1 | CP2 | CP3 | B&C1 | B&C2 |
| 10-0.3-5 | 0.00 | 0.00 | 0.00 | 1.80 | 3.80 | 0.80 | 0.80 | 0.80 | 4.60 | 7.00 | 0.00 | 0.00 | 0.00 | 1.60 | 5.80 |
| 10-0.3-10 | 0.60 | 0.60 | 0.60 | 7.00 | 13.60 | 5.40 | 6.40 | 6.40 | 18.20 | 21.00 | 2.60 | 3.40 | 3.40 | 12.20 | 15.00 |
| 10-0.3-15 | 2.00 | 3.40 | 3.40 | 15.80 | 29.80 | 6.80 | 8.60 | 8.60 | 28.20 | 30.20 | 8.20 | 10.20 | 10.00 | 33.00 | 34.40 |
| 10-0.5-5 | 0.80 | 1.80 | 1.80 | 3.20 | 6.80 | 0.80 | 0.80 | 0.80 | 5.20 | 8.40 | 0.00 | 0.00 | 0.00 | 2.00 | 3.40 |
| 10-0.5-10 | 4.80 | 7.80 | 7.40 | 20.40 | 24.80 | 10.00 | 11.20 | 11.20 | 34.40 | 48.00 | 5.80 | 8.60 | 8.20 | 31.40 | 27.20 |
| 10-0.5-15 | 8.60 | 11.20 | 11.20 | 39.60 | 67.80 | 28.00 | 29.00 | 29.00 | 87.80 | 102.60 | 17.60 | 30.60 | 30.80 | 55.80 | 103.40 |
| 10-0.7-5 | 0.20 | 0.20 | 0.20 | 3.00 | 3.40 | 0.40 | 0.60 | 0.60 | 5.00 | 10.20 | 1.60 | 2.20 | 2.20 | 6.40 | 9.60 |
| 10-0.7 -10 | 2.40 | 6.20 | 6.40 | 23.40 | 32.80 | 17.40 | 24.80 | 25.20 | 69.20 | 70.40 | 8.00 | 10.40 | 10.40 | 33.80 | 61.40 |
| 10-0.7-15 | 4.60 | 7.40 | 8.40 | 22.60 | 38.00 | 49.20 | 95.40 | 94.80 | 209.80 | 342.40 | 29.00 | 54.20 | 53.60 | 90.80 | 266.00 |
| 20-0.3-10 | 1.80 | 2.20 | 2.20 | 16.40 | 17.20 | 3.00 | 4.80 | 4.80 | 22.00 | 22.80 | 3.60 | 3.40 | 3.60 | 13.80 | 24.40 |
| 20-0.3-20 | 17.00 | 28.20 | 54.60 | 118.60 | 259.40 | 65.00 | 127.40 | 124.00 | 293.20 | 812.60 | 59.40 | 119.60 | 123.60 | 273.40 | 421.20 |
| 20-0.3-30 | 28.60 | 63.20 | 62.60 | 145.40 | 586.80 | 178.20 | 271.20 | 285.40 | 1131.40 | 6726.80 | 107.80 | 183.00 | 186.40 | 460.20 | 911.20 |
| 20-0.5-10 | 1.60 | 2.40 | 2.40 | 11.80 | 22.00 | 5.20 | 6.00 | 6.00 | 32.80 | 25.00 | 4.60 | 7.60 | 7.80 | 17.80 | 28.40 |
| 20-0.5-20 | 15.00 | 31.60 | 30.40 | 104.20 | 135.20 | 172.20 | 233.00 | 184.40 | 317.67 | 665.33 | 39.40 | 74.40 | 76.00 | 167.00 | 727.60 |
| 20-0.5-30 | 50.80 | 126.20 | 129.00 | 296.40 | 2586.80 | 109.40 | 160.60 | 266.40 | 1248.40 | 7362.20 | 128.60 | 221.60 | 239.60 | 671.00 | 6853.60 |
| 20-0.7-10 | 1.40 | 2.40 | 2.40 | 9.80 | 16.80 | 4.20 | 4.80 | 4.80 | 19.40 | 25.80 | 0.60 | 0.60 | 0.60 | 13.80 | 16.60 |
| 20-0.7-20 | 30.60 | 65.60 | 74.80 | 112.60 | 299.00 | 167.80 | 245.20 | 184.00 | 879.75 | 6182.75 | 65.20 | 108.00 | 108.80 | 179.00 | 794.20 |
| 20-0.7-30 | 52.20 | 134.60 | 137.80 | 388.40 | 1765.00 | 217.00 | 255.40 | 271.40 | 2170.20 | 14154.60 | 130.60 | 207.60 | 218.00 | 675.40 | 5977.60 |
| 30-0.3-15 | 4.00 | 6.00 | 6.00 | 26.40 | 33.60 | 19.20 | 33.20 | 32.80 | 127.60 | 120.40 | 10.60 | 13.80 | 13.20 | 52.20 | 79.20 |
| 30-0.3-30 | 32.60 | 75.80 | 78.20 | 212.80 | 451.80 | - | - | - | - | - | - | - | - | - | - |
| 30-0.5-15 | 3.40 | 6.20 | 6.20 | 32.80 | 36.40 | 11.60 | 12.40 | 12.80 | 59.00 | 43.80 | 7.80 | 9.00 | 9.00 | 35.80 | 39.80 |
| 30-0.7-15 | 14.60 | 19.80 | 19.80 | 63.00 | 55.00 | 26.20 | 45.20 | 46.00 | 136.40 | 138.80 | 2.80 | 3.20 | 2.80 | 14.80 | 24.80 |
| Average | 12.62 | 27.40 | 29.35 | 76.15 | 294.81 | 52.28 | 75.09 | 76.20 | 328.58 | 1758.15 | 30.18 | 51.02 | 52.76 | 135.30 | 782.13 |

Table 3.8: Number of separation problems solved by the iterative cutting plane and the Branch-and-Cut algorithms

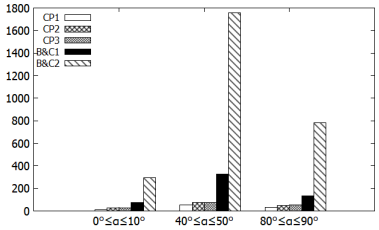
| Angle | $0° \leq \alpha \leq 10°$ | | | | | $40° \leq \alpha \leq 50°$ | | | | | $80° \leq \alpha \leq 90°$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ - $d$ - $K$ | CP1 | CP2 | CP3 | B&C1 | B&C2 | CP1 | CP2 | CP3 | B&C1 | B&C2 | CP1 | CP2 | CP3 | B&C1 | B&C2 |
| 10-0.3-5 | 1.00 | 1.00 | 1.00 | 3.00 | 3.60 | 1.40 | 1.40 | 1.40 | 4.20 | 4.80 | 1.00 | 1.00 | 1.00 | 3.00 | 4.00 |
| 10-0.3-10 | 1.40 | 1.40 | 1.40 | 3.60 | 4.60 | 2.60 | 3.00 | 3.00 | 7.40 | 6.40 | 2.40 | 2.60 | 2.60 | 6.20 | 7.20 |
| 10-0.3-15 | 1.80 | 2.40 | 2.40 | 6.20 | 6.60 | 3.40 | 4.40 | 4.40 | 8.80 | 10.20 | 3.00 | 3.60 | 3.60 | 7.80 | 7.80 |
| 10-0.5-5 | 1.80 | 2.80 | 2.80 | 4.20 | 6.00 | 1.40 | 1.40 | 1.40 | 5.00 | 5.80 | 1.00 | 1.00 | 1.00 | 3.00 | 3.20 |
| 10-0.5-10 | 2.80 | 4.40 | 4.20 | 9.60 | 11.80 | 4.40 | 4.80 | 4.80 | 11.20 | 14.20 | 3.40 | 4.80 | 4.60 | 10.60 | 9.80 |
| 10-0.5-15 | 3.40 | 4.40 | 4.20 | 10.80 | 22.80 | 7.00 | 9.00 | 9.00 | 17.80 | 20.40 | 6.00 | 10.00 | 10.00 | 14.40 | 28.40 |
| 10-0.7-5 | 1.20 | 1.20 | 1.20 | 4.40 | 3.40 | 1.40 | 1.60 | 1.60 | 5.20 | 5.40 | 1.80 | 2.20 | 2.20 | 5.00 | 6.00 |
| 10-0.7-10 | 2.60 | 4.60 | 4.60 | 10.40 | 12.40 | 7.40 | 11.00 | 10.80 | 22.80 | 22.60 | 4.20 | 4.80 | 4.80 | 11.20 | 17.80 |
| 10-0.7-15 | 3.00 | 4.00 | 4.60 | 7.60 | 11.00 | 11.00 | 19.40 | 19.20 | 40.00 | 69.60 | 8.40 | 13.00 | 12.40 | 19.20 | 55.80 |
| 20-0.3-10 | 2.00 | 2.00 | 2.00 | 6.40 | 5.60 | 2.60 | 3.60 | 3.60 | 9.60 | 7.60 | 2.80 | 3.00 | 3.00 | 6.20 | 7.60 |
| 20-0.3-20 | 6.20 | 9.60 | 18.00 | 23.00 | 64.00 | 13.80 | 26.40 | 26.00 | 47.60 | 138.00 | 13.60 | 30.40 | 31.60 | 49.60 | 97.40 |
| 20-0.3-30 | 6.40 | 15.40 | 14.80 | 23.80 | 106.80 | 17.40 | 21.60 | 23.60 | 105.20 | 722.40 | 14.40 | 23.40 | 23.60 | 49.60 | 106.40 |
| 20-0.5-10 | 2.40 | 2.80 | 2.80 | 6.80 | 7.20 | 3.20 | 3.80 | 3.80 | 13.00 | 8.00 | 3.60 | 5.20 | 5.40 | 10.60 | 10.80 |
| 20-0.5-20 | 5.80 | 9.20 | 8.40 | 22.80 | 27.20 | 24.60 | 47.20 | 33.20 | 77.60 | 361.00 | 10.00 | 19.60 | 19.60 | 28.20 | 110.60 |
| 20-0.5-30 | 9.20 | 18.80 | 18.80 | 43.20 | 301.60 | 15.67 | 19.00 | 20.80 | 76.00 | 471.00 | 16.60 | 24.20 | 26.00 | 66.60 | 829.00 |
| 20-0.7-10 | 1.80 | 2.20 | 2.20 | 6.00 | 5.80 | 3.20 | 3.40 | 3.40 | 8.20 | 9.40 | 1.60 | 1.60 | 1.60 | 6.60 | 5.20 |
| 20-0.7-20 | 10.40 | 25.00 | 28.80 | 31.80 | 106.00 | 26.40 | 44.80 | 26.60 | 110.25 | 655.25 | 15.40 | 23.00 | 23.40 | 29.60 | 138.80 |
| 20-0.7-30 | 9.20 | 17.40 | 18.00 | 60.00 | 222.60 | 15.00 | 15.80 | 16.80 | 156.40 | 1096.20 | 13.20 | 17.80 | 18.60 | 67.00 | 622.80 |
| 30-0.3-15 | 2.80 | 3.80 | 3.80 | 8.80 | 8.00 | 8.80 | 13.80 | 13.60 | 39.80 | 33.00 | 5.20 | 7.60 | 7.40 | 16.80 | 31.60 |
| 30-0.3-30 | 8.80 | 17.40 | 18.20 | 35.00 | 81.80 | - | - | - | - | - | - | - | - | - | - |
| 30-0.5-15 | 2.40 | 3.80 | 3.80 | 13.40 | 10.20 | 6.40 | 7.00 | 7.00 | 15.40 | 9.60 | 5.00 | 5.40 | 5.40 | 12.60 | 9.40 |
| 30-0.5-30 | 11.40 | 18.20 | 18.40 | 54.80 | 458.60 | - | - | - | - | - | - | - | - | - | - |
| 30-0.7-15 | 6.60 | 8.60 | 8.60 | 24.00 | 17.60 | 9.00 | 15.80 | 16.00 | 34.80 | 38.00 | 2.60 | 3.00 | 2.80 | 8.00 | 7.60 |
| 30-0.7-30 | 11.20 | 19.00 | 19.60 | 47.20 | 480.60 | - | - | - | - | - | - | - | - | - | - |
| Average | 4.82 | 8.31 | 8.86 | 19.45 | 82.74 | 8.86 | 13.25 | 11.90 | 38.87 | 176.61 | 6.44 | 9.87 | 10.03 | 20.56 | 100.82 |



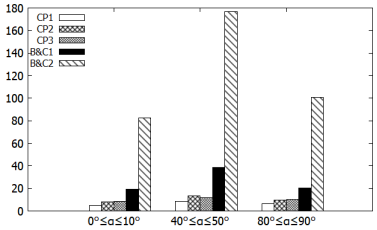

(a) Gap between the optimal solution and the linear relaxation




(b) percentage of CPU-SP/CPU-total




(c) Number of cuts generated by **CP** and **B&C** algorithms




(d) Number of separation problems solved

Figure 3.2: Additional statistics for the algorithms and formulations

generated by each algorithm. As we can see in Figure 3.2c, the **CP1** algorithm generates a smaller number of cuts to obtain the optimal solution than the other iterative cutting plane and branch-and-cut algorithms. Specifically, **B&C1** (resp. **B&C2**) generates more than six (resp. twenty) times the number of cuts of **CP1**. In addition, we can remark that the similar performances of **CP1** and **CP2** can be partly explained by the equivalence between the two valid inequalities used in these algorithms. The same remark can be shown for the number of iterations in Figure 3.2d.

To evaluate the performance of the different formulations, a performance profile [53] of solution time on the random instances is given in Figure 3.3. The chart represents the proportion of instances for which each algorithm is not more than $x$-times worst than the best algorithm. For example, if we take $x = 1$, we can see that **CP3** is the best algorithm for 40% of instances, while **CP1** has the best CPU-time for more than 20% of instances.

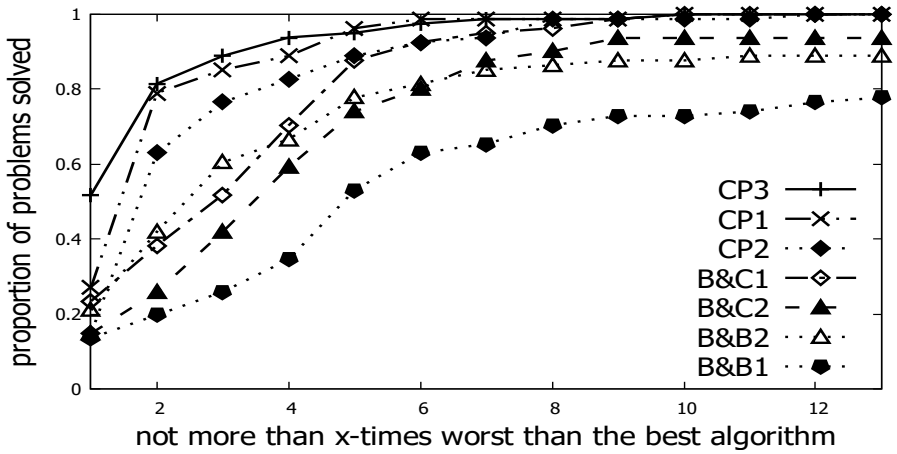



Figure 3.3: Performance profile comparing the different algorithms for random instances.

Table 3.9 sums up the number of instances solved to optimality in 3600$s$. The iterative cutting plane approaches **CP1** and **CP2** are similar to each other. They can solve up to 19% (resp. 8%) more instances than the compact formulations (resp. the branch-and-cut algorithms). Also, **CP3** solves more instances than the other methods.

Table 3.9: Number of random instances solved to optimality

| | B&B1 | B&B2 | CP1 | CP2 | CP3 | B&C1 | B&C2 |
|---|---|---|---|---|---|---|---|
| $0° \leq \alpha \leq 10°$ | 100 | 109 | 120 | 120 | 120 | 110 | 109 |
| $40° \leq \alpha \leq 50°$ | 88 | 89 | 104 | 104 | 120 | 94 | 92 |
| $80° \leq \alpha \leq 90°$ | 88 | 97 | 105 | 105 | 120 | 102 | 102 |
| Sum | 276 | 295 | 329 | 329 | 360 | 306 | 303 |

### 3.5.2 Real instances

We apply the proposed algorithms to solve FCNDP-SPC on real instances from the literature representing the road networks of Ravenna (Italy) and Albany, NY (USA). In these instances, the fixed costs are equal to zero.

#### 3.5.2.1 Ravenna data

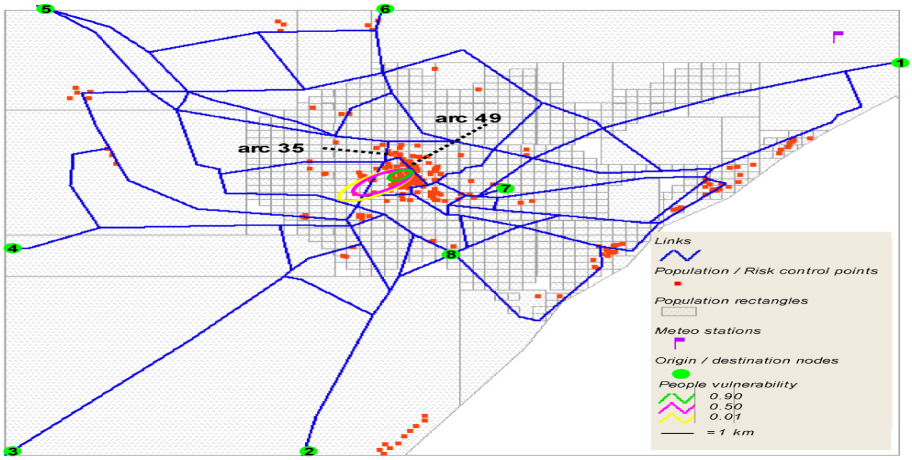



Figure 3.4: Hazmat transportation network of Ravenna, Italy.

The area of Ravenna, Italy (Figure 3.4, [23]), measures $28.8km \times 26km$ [23]. To describe its road network, 111 nodes and 143 edges are considered. There are 8 nodes with a transportation requirement for 4 hazmats: LPG, Methanol, Gasoline, and Chlorine. The 8 nodes form 35 origin-destination (O-D) pairs, and the number of commodities to be transported between each O-D varies between 16 and 29684. The variable cost associated to each edge is measured using population density. Seven instances are taken from the original Ravenna data with different number of commodities $K \in \{5, 10, 15, 20, 25, 30, 35\}$, where, in each case, we take the first $K$ pairs of origin-destination. The results obtained after applying the six models on these instances are displayed in Table 3.10.

The presented results show that the iterative cutting plane algorithms are faster, in average, than the other algorithms and are able to solve more instances to optimality. Also, our approach **CP1** is more efficient for this set of instances.

Unlike the other instances, in Ravenna data, many commodities have the same origin and the same destination. For this reason, the inclusion of the valid inequality (3.10) as a constraint in the initial model can improve the results.

Table 3.10: Comparison of CPU time in seconds on the Ravenna data

| K | B&B1 | B&B2 | CP1 | CP2 | CP3 | B&C1 | B&C2 |
|---|---|---|---|---|---|---|---|
| 5 | 41.91 | 0.45 | 1.59 | 1.52 | 1.52 | 2.32 | 2.45 |
| 10 | 145.90 | 0.98 | 3.15 | 3.09 | 2.98 | 3.29 | 4.68 |
| 15 | 1,169.96 | 3.80 | 33.01 | 23.13 | 27.88 | 50.75 | 96.00 |
| 20 | - | 8.98 | 51.61 | 47.93 | 46.21 | 78.13 | 2,896.65 |
| 25 | - | 259.91 | 82.74 | 86.72 | 63.15 | - | - |
| 30 | - | 1,494.56 | 151.28 | 283.88 | 265.86 | - | - |
| 35 | - | - | 222.11 | 451.68 | 351.18 | - | - |
| Average | - | - | 77.93 | 128.28 | 108.40 | - | - |

Table 3.11 presents the CPU time obtained after adding the valid inequality. According to the results in Table 3.11, the addition of valid inequality (3.10)

Table 3.11: CPU time in seconds on the Ravenna data with additional valid inequality

| K | B&B1 | B&B2 | CP1 | CP2 | CP3 | B&C1 | B&C2 |
|---|---|---|---|---|---|---|---|
| 5 | 41.91 | 0.45 | 1.75 | 1.66 | 1.53 | 1.59 | 1.57 |
| 10 | 160.03 | 1.10 | 3.49 | 3.61 | 3.09 | 2.32 | 2.38 |
| 15 | 585.17 | 2.43 | 5.94 | 6.40 | 5.34 | 3.66 | 3.92 |
| 20 | - | 8.98 | 70.90 | 42.98 | 35.28 | 205.94 | 35.47 |
| 25 | - | 151.27 | 63.50 | 55.25 | 43.95 | 105.33 | 355.98 |
| 30 | - | 371.25 | 57.83 | 105.59 | 66.78 | 156.80 | 290.37 |
| 35 | - | - | 84.20 | 174.00 | 119.83 | 229.43 | 414.51 |
| Average | - | - | 36.63 | 55.64 | 39.40 | 100.72 | 157.74 |

reduces the running time for all algorithms. Comparing Tables 3.10 and 3.11, we see that the branch-and-cut algorithms solved more instances to optimality when we include inequalities (3.10). Also, the results show that the CPU times for all the algorithms with the constraint (3.10) is about 2.25 times faster than the CPU of algorithms without it.

#### 3.5.2.2 Albany data

The Albany data set is composed of information on the highway system of Albany, NY, USA (Figure 3.5), used for the routing of Hazmat shipments problem [131]. The highway system is represented by a network of 90 nodes and 149 edges.

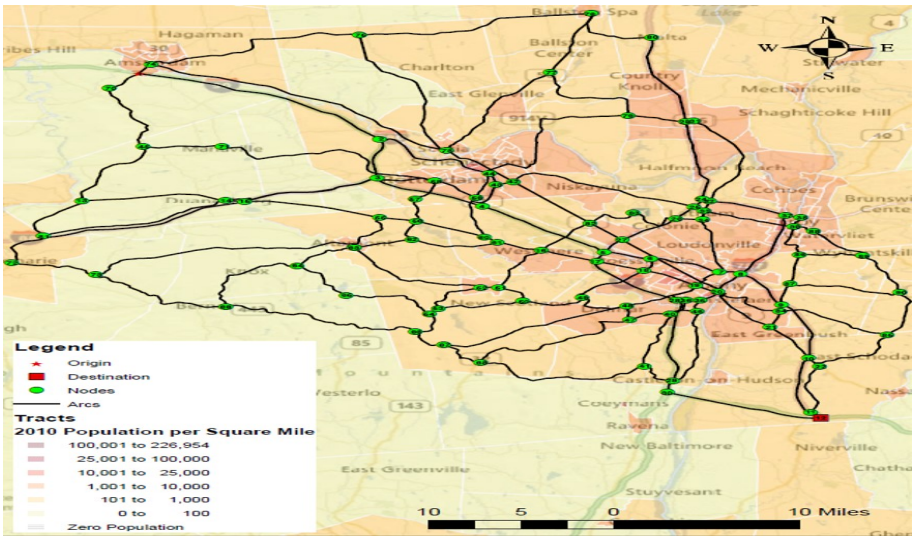



Figure 3.5: Hazmat transportation network of Albany, USA.

Table 3.12: Comparison of CPU time on the Albany data

| K | B&B1 | B&B2 | CP1 | CP2 | CP3 | B&C1 | B&C2 |
|---|---|---|---|---|---|---|---|
| 5 | 83.97 | 0.65 | 23.43 | 37.06 | 20.13 | 5.65 | 2.71 |
| 10 | - | 919.53 | 87.01 | - | 996.03 | 332.79 | - |
| 15 | - | - | 193.51 | - | - | - | - |
| 20 | - | - | 255.40 | - | - | - | - |
| 25 | - | - | 652.84 | - | - | - | - |

We generated a set of origin-destination pairs for each shipment. The number of commodities takes a value in the set $K = \{5, 10, 15, 20, 25\}$, and the demand for each O-D is generated uniformly in $[1, 100]$. Table 3.12 displays the results obtained by each method.

According to the results in Table 3.12, only **CP1** is able to find the optimal solution for up to 25 different commodities.

# Chapter 4

# Fixed Charge Network Design Problem with User Optimal Flows

In recent years, green networking has attracted a lot of attention from device manufacturers and Internet Service Providers (ISP) to reduce the energy consumption. In the literature, the energy-aware traffic engineering problem is proposed to minimize the total energy consumption by switching off unused network devices (routers and links) while guaranteeing full network connectivity.

In this chapter, we are interested in the problem of energy-aware Traffic Engineering while using multi-path routing (ETE-MPR) to minimize link capacity utilization in ISP backbone networks. To this end, we propose a bi-level optimization model where the upper level represents the energy management function and the lower level refers to the deployed multi-path routing protocol. Then, we reformulate it as a one-level MILP replacing the second level problem by different sets of optimality conditions. We further use these formulations to solve the problem with compact formulations, cutting plane and branch-and-cut algorithms. The

computational experiments are performed on real instances to compare the proposed algorithms and to evaluate the efficiency of our model against the existing single-path and multi-objective approaches.

## 4.1 Introduction

As we are entering the area of Internet of Everything, the number of devices connected to the IP networks will increase by one and a half times the global population (8 billion) by 2022, says the latest Cisco Visual Networking Index (VNI) [42]. Consequently, the monthly global IP traffic will be 77 exabytes by 2022, and the annual traffic will reach almost one zettabyte [42]. With the tremendous growth of Internet traffic, the network energy consumption is inherently growing fast with a rate of 10% per year, which exceeded 350 TWh and represented 1.8% of the worldwide electricity consumption in 2012 [92]. It is even reported that communication networks will consume as much as 51% of the global electricity in the worst-case by 2030, if its energy efficiency is not improved enough [8]. Thus, the problem of energy efficiency is becoming critical for communication networks nowadays.

To reduce energy consumption, green networking has attracted a lot of attention from device manufacturers and Internet Service Providers (ISP). To this end, many technologies and approaches have been proposed by the networking community to cut the carbon footprint of the ISP backbone networks [2; 6; 22; 38; 46; 78]. Among them, one promising solution is the energy management of network elements including routers and communication links [2; 6; 38; 41]. In fact, network resources like processing power and memory are oversized in communication networks nowadays, which results in a low utilization of 30-40% in low traffic periods [46]. As a result, important energy savings can be obtained by switching into the

sleeping mode the network elements which are not used for the data delivery during a certain period. Following this idea, authors in [38] tried to minimize the total energy consumption by switching off unused network devices while guaranteeing full network connectivity. In [6], the authors studied the energy-aware traffic engineering problem subject to elastic traffic and max-min fair bandwidth allocation. The work in [2] is focused on the energy minimization strategy that selectively switches off devices according to the traffic level. The authors of [41] proposed an OSPF-integrated routing strategy for QoS-aware energy saving in backbone networks.

However, all the work mentioned above only considered the single path routing (using protocols like Routing Information Protocol (RIP) or Open Shortest Path First (OSPF) protocol) for the energy-aware traffic engineering [2; 6; 38; 41], where only one path is used per communication request to deliver data from its source node to its destination. The single path routing strategy may lead to a high blocking probability when some network links are congested during the period of high traffic load. In light of this, multi-path routing has been investigated [13; 36; 39; 99; 104; 127; 142].

Recent works show that Internet-wide multi-path routing is feasible and very attractive: First, utilizing diverse paths through multi-path routing enables to reduce blocking probability by load balancing, which permits to improve both the utilization efficiency of network bandwidth and the transmission reliability. Second, a higher Quality of Service (QoS) and Quality of Experience (QoE) can be obtained [95; 99; 127]. Thus, many multi-path routing protocols and algorithms have been proposed, for instance, generalized destination-based multi-path routing [142], equal-cost multi-path routing [39] and minimum routing cost multi-path routing [104]. In general, there are two main variants of multi-path routing, whose optimization goal is either to minimize the network congestion ratio through load

balancing [13; 80; 111; 142], or to minimize the total routing cost of traffic flows on all paths [36; 39; 104].

In this work, we deal with the problem of energy-aware traffic engineering while using multi-path routing to minimize link capacity utilization in ISP backbone networks. We suppose the network control plane disposes of several key functions like energy management, routing, and signaling. To achieve energy-efficient, we suppose that the energy management function and the deployed routing protocol work independently. On the one hand, to improve the energy efficiency, the energy management function of the ISP network tends to minimize the network energy consumption by putting on sleep unused network devices according to the actual traffic demands. On the other hand, the network operator should also assure that the deployed routing protocol is able to provide all the traffic demands in the link capacity constraint. We assume that the multi-path routing protocol minimizing total link capacity utilization (a simple version of total routing cost of all traffic flows on paths) [36; 39; 104] is used by the ISP network operator. However, the energy consumption and link capacity utilization cannot be optimized simultaneously, and an equilibrium should be achieved for the two control plane functions. Different from the literature, we propose a bi-level formulation for this green network optimization problem instead of the traditional single-level one [2].

Our bi-level formulation is motivated by the following two reasons. First, from the point of view of the ISP network operator, it is difficult to tune the weights of two network resources (*i.e.*, energy consumption and link capacity utilization) so as to find a good trade-off. In contrast, our formulation enables to find an equilibrium between the two network resources utilization, which is transparent for the ISP network operator. Second, instead of developing new energy-aware routing protocols, we can take advantage of the existing work by

deploying directly the proposed multi-path routing protocols in [36; 39; 104], which minimize the total link capacity utilization. In our bi-level model, the upper level represents the energy management function aiming at cutting the total network energy consumption. The lower level refers to the deployed multi-path routing protocol, whose objective is to reduce the total link capacity utilization.

## 4.2 Problem definition and notation

We consider an ISP backbone network (Figure 4.1) with slow dynamics, where the traffic demand is supposed to be relatively stable during a period of time. With the help of intelligent network management like Software Defined Networking (SDN), the control plane of the ISP network disposes of key functions like energy management and routing. This kind of network management permits us to change the state of network devices and engineer the traffic in a centralized way.

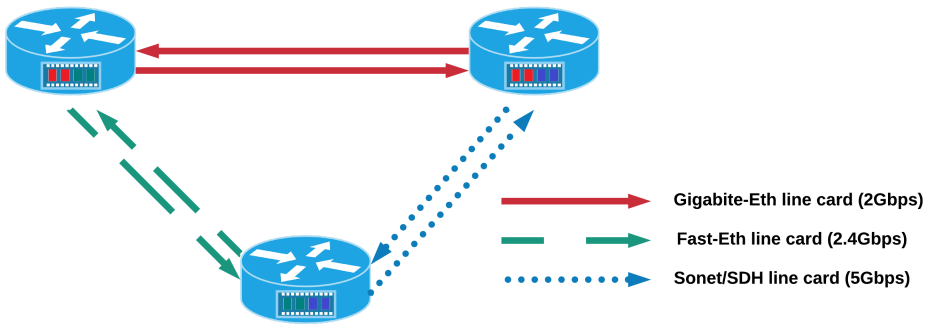



Figure 4.1: ISP model architecture.

To correctly model the energy consumption of a router, we should get familiar with its architecture. We consider the widely used router architecture, which consists of a chassis and a set of line cards [2; 6]. The chassis is used to provide computation and switching functionalities. One example can be chassis Junifer

M10i. Line cards are usually used to provide communication interfaces and network processing. To achieve power savings, the router elements (chassis and line cards) can be powered on and off by the control plane energy management function. Once activated, the chassis has a constant energy consumption independent from the amount of traffic passing through the router. It should be noted that the chassis can only be put into sleep when all of its line cards are OFF. To satisfy heterogeneous traffic, a router can be plugged with different kinds of line cards, which result in distinct energy efficiencies. For instance, we have line cards like Gigabit-Eth (2Gbps), Fast-Eth 12 ports (2.4Gbps), SONET /SDH OC-48c (5Gbps), among others, whose capacity and hourly power consumption are listed in [2; 6]. However, a communication link must be connected to a same type of line cards available on the two ending routers. Thus, each communication link is accordingly associated with a capacity and an energy-efficiency factor, which depends on the types of line cards used. For energy savings, only a subset of line cards are powered on to assure just enough bandwidth for the carried traffic while the rest will be put to sleep. For simplicity reasons, we thus assume the energy consumption on communication links is proportional to its carried traffic rate.

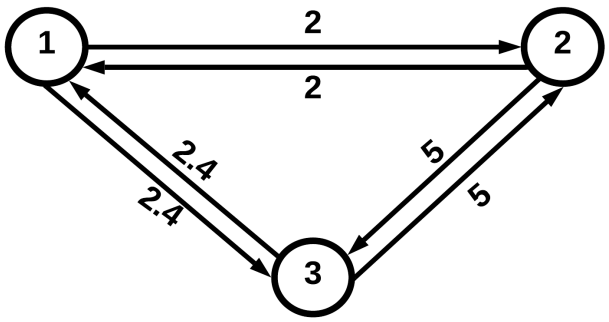



Figure 4.2: The graph representation of the ISP network (Figure ).

We consider the ISP backbone network modeled by the bi-directed graph

$G(V, A)$ (Figure 4.2), where $V$ is the set of routers and $A$ represents the set of communication links. Since the backbone networks are bi-directed and for modelling the activation link constraints (Section 4.3), we define the set of (undirected) edges associate to $A$ as $E = \{\{i, j\} : (i, j) \in A \text{ or } (j, i) \in A\}$.

Let $C_{ij} \geq 0$ denotes the capacity of the link $(i, j) \in A$ defined by the line cards on routers $i$ and $j$, and $K$ be the set of traffic demands. Demand $k \in K$ must send a volume of $\phi^k$ from node $o(k)$ to node $d(k)$.We denote by $P_i$ the energy consumption of the chassis in router $i$, and we use $g_{ij} = g_{ji}$ to denote the energy efficiency of the line cards connecting the link $(i, j) \in A$.

The energy-aware traffic engineering with multi-path routing problem (ETE-MPR) problem consists to select a set of routers (nodes) and links (edges) to be activated with minimum energy consumption. Assuming that each traffic demand uses a multi-path flow minimizing the total link capacity utilization.

## 4.3 Mathematical formulations

In this section, we will introduce four different formulations for the ETE-MPR problem. We start by a bi-level integer programming model in Subsection 5.3.1, then, the model will be transformed into three one-level MILP formulations (Subsections 4.3.2, 4.3.3 and 4.3.4). For comparison purpose, the Subsection 4.3.5 will be devoted to present the unsplittable flow version of the energy-aware traffic problem.

### 4.3.1 Bi-level formulation

The problem is naturally cast as a bi-level optimization problem where the upper level represents the energy management function and the lower level refers to the deployed multi-path routing protocol. For each $i \in V$ and $e(i, j) = \{i, j\} \in$

$E$, we introduce the binary variables $z_i$ and $y_e$ that represent the power status (ON/OFF) of router $i$ and link $e = \{i, j\}$, respectively. Furthermore, for each $k \in K$ and $(i, j) \in A$ let the continuous positive variable $x_{ij}^k$ be the flow on $(i, j)$ of demand $k$.

The problem belongs to the class of NP-hard problems and can be modeled as a bi-level mixed integer programming problem, as follows:

$$\min \quad \sum_{i \in V} P_i z_i + \sum_{(i,j) \in A} \sum_{k \in K} g_{ij} x_{ij}^k \tag{4.1a}$$

$$\text{s.t.} \quad z_i \in \{0, 1\}, \qquad \forall i \in V, \tag{4.1b}$$

$$y_e \geq z_i + z_j - 1, \qquad \forall e = \{i, j\} \in E, \tag{4.1c}$$

$$y_e \leq z_i, \qquad \forall e = \{i, j\} \in E, \tag{4.1d}$$

$$y_e \leq z_j, \qquad \forall e = \{i, j\} \in E, \tag{4.1e}$$

$$\min \quad \sum_{k \in K} \sum_{(i,j) \in A} \frac{1}{C_{ij}} x_{ij}^k \tag{4.1f}$$

$$\text{s.t.} \quad \sum_{(i,j) \in \delta^+(i)} x_{ij}^k - \sum_{(i,j) \in \delta^-(i)} x_{ji}^k = b_i^k, \qquad \forall i \in V, \forall k \in K, \tag{4.1g}$$

$$\sum_{k \in K} x_{ij}^k \leq C_{ij} y_{e(i,j)}, \qquad \forall (i, j) \in A, \tag{4.1h}$$

$$x_{ij}^k \geq 0, \qquad \forall (i, j) \in A, \forall k \in K. \tag{4.1i}$$

where:

$$b_i^k = \begin{cases} \phi^k, & \text{if } \ i = o(k), \\ -\phi^k, & \text{if } \ i = d(k), \\ 0, & \text{otherwise.} \end{cases}$$

The upper-level objective function (4.1a) aims to minimize the energy consumption of routers and links, while the objective of the second level problem is to

minimize the total link capacity utilization (5.4e). Constraints (4.1c)-(4.1e) ensure that a link $(i, j)$ is activated only if the two routers $i$ and $j$ are switched ON. Constraints (4.1g) and (4.1h) define the classical flow and capacity constraints, respectively. While (4.1i) requires the variables $x_{ij}^k$ to be non-negative.

### 4.3.2 One level formulation

The problem can be reformulated as a one-level integer programming problem by replacing the second level problem by the associated optimality conditions. Following the classical reformulation used in [56; 109**?** ], among others, we can apply the fundamental theorem of duality and complementarity slackness conditions [16] to replace the follower problem:

$$\min \quad \sum_{i \in V} P_i z_i + \sum_{(i,j) \in A} \sum_{k \in K} g_{ij} x_{ij}^k$$

$$\text{s.t.} \quad (4.1b) - (4.1e),$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij}^k - \sum_{(i,j) \in \delta^-(i)} x_{ji}^k = b_i^k, \qquad \forall i \in V, \forall k \in K, \tag{4.2a}$$

$$(C_{ij} y_{e(i,j)} - \sum_{k \in K} x_{ij}^k) W_{ij} = 0, \qquad \forall (i,j) \in A, \forall k \in K, \tag{4.2b}$$

$$(\frac{1}{C_{ij}} + \pi_i^k - \pi_j^k + W_{ij}) x_{ij}^k = 0, \qquad \forall (i,j) \in A, \forall k \in K, \tag{4.2c}$$

$$\frac{1}{C_{ij}} + \pi_i^k - \pi_j^k + W_{ij} \geq 0, \qquad \forall (i,j) \in A, \forall k \in K, \tag{4.2d}$$

$$\sum_{k \in K} x_{ij}^k \leq C_{ij} y_{e(i,j)}, \qquad \forall (i,j) \in A, \tag{4.2e}$$

$$W_{ij} \geq 0, \qquad \forall (i,j) \in A, \tag{4.2f}$$

$$\pi_i^k \in \mathbb{R}, \qquad \forall i \in V, \forall k \in K, \tag{4.2g}$$

$$x_{ij}^k \geq 0, \qquad \forall (i,j) \in A, \forall k \in K. \tag{4.2h}$$

This formulation contains non-linear constraints (4.2b) and (4.2c). We can linearize them using the big-M method by adding a new binary variable for each constraint. After this modification, we obtain a one-level mixed integer programming reformulation for (4.1), as follows:

$$\min \quad \sum_{i \in V} P_i z_i + \sum_{(i,j) \in A} \sum_{k \in K} g_{ij} x_{ij}^k$$

$$\text{s.t.} \quad (4.1b) - (4.1e), (4.2d) - (4.10i)$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij}^k - \sum_{(i,j) \in \delta^-(i)} x_{ji}^k = b_i^k, \qquad \forall i \in V, \forall k \in K, \tag{4.3a}$$

$$C_{ij} y_{ee(i,j)} - \sum_{k \in K} x_{ij}^k \leq M \eta_{ij}, \qquad \forall (i,j) \in A, \tag{4.3b}$$

$$M \eta_{ij} + W_{ij} \leq M, \qquad \forall (i,j) \in A, \tag{4.3c}$$

$$\frac{1}{C_{ij}} + \pi_i^k - \pi_j^k + W_{ij} \leq M \mu_{ij}^k, \qquad \forall (i,j) \in A, \forall k \in K, \tag{4.3d}$$

$$M \mu_{ij}^k + x_{ij}^k \leq M, \qquad \forall (i,j) \in A, \forall k \in K, \tag{4.3e}$$

$$\eta_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A, \tag{4.3f}$$

$$\mu_{ij}^k \in \{0,1\}, \qquad \forall (i,j) \in A, \forall k \in K. \tag{4.3g}$$

where $M$ is a precomputed large number. Unlike the other constraints, where the value of $M$ cannot be bounded, in constraint (4.3b) $M$ can be replaced by a lower bound $M = C_{ij}$ for each $(i,j) \in A$.

### 4.3.3 BILP formulation based on flow constraints

We propose an alternative one level formulation to the problem where the second level problem is replaced by a set of constraints eliminating unfeasible flows. Let

us denote by $\mathcal{F}$ the set of all feasible flows defined as:

$$\mathcal{F} = \{F = (x, y, z) : (4.1b) - (4.1e), (4.1g) - (4.1i)\}$$

For each $F \in \mathcal{F}$, we also define $C(F) = \sum_{k \in K} \sum_{(i,j) \in A} \frac{1}{C_{ij}} x_{ij}^k$ as the total capacity link utilization of a flow $F$ and $G(F) = \{e \in E : y_e = 1\}$ as the associated set of activated edges.

The flow $F = (x, y, z)$ is bi-level unfeasible if there exists a solution $F' = (x', y, z) \in \mathcal{F}$ such that:

$$C(F') < C(F).$$

Hence, the bi-level problem can be reformulated as:

$$\min \quad \sum_{i \in V} P_i z_i + \sum_{(i,j) \in A} \sum_{k \in K} g_{ij} x_{ij}^k$$

$$\text{s.t.} \quad (4.1b) - (4.1e)$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij}^k - \sum_{(i,j) \in \delta^-(i)} x_{ji}^k = b_i^k, \qquad \forall i \in V, \forall k \in K, \tag{4.4a}$$

$$\sum_{k \in K} \sum_{(i,j) \in A} \frac{1}{C_{ij}} x_{ij}^k \leq C(F) + (|G(F)| - \sum_{e \in G(F)} y_e) M, \qquad \forall F \in \mathcal{F}, \tag{4.4b}$$

$$\sum_{k \in K} x_{ij}^k \leq C_{ij} y_{e(i,j)}, \qquad \forall (i,j) \in A, \tag{4.4c}$$

$$x_{ij}^k \geq 0, \qquad \forall (i,j) \in A, \forall k \in K. \tag{4.4d}$$

Constraint (4.4b) is in charge of eliminating all bi-level unfeasible flows $F$ on

the set of activated links $\{e = \{i, j\} \in E : \quad y_e = 1\}$. Again, $M$ is a precomputed large number. The above formulation contains a polynomial number of variables and an exponential number of constraints.

### 4.3.4 Residual network optimality conditions

We can remark that the follower problem is a particular case of the minimum cost flow problem where the cost of each arc $(i, j)$ is equal to $\frac{1}{C_{ij}}$. In this subsection, we propose to reformulate this problem by using a set of optimality conditions for the min-cost flow problems defined on the residual network (generalization of the one-commodity minimum cost flow optimality conditions presented in Section 2.3).

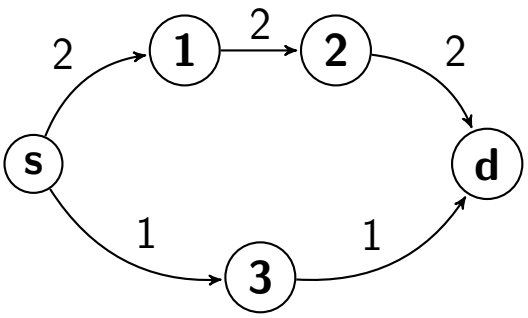
We start by giving the definition of the residual network and the associated optimality conditions. Let $x^k = (x^k_{ij})_{(i,j)\in A} \in \mathbb{R}^{|A|}$ be the flow between the origin-destination pair of demand $k \in K$. Notice that $x = (x^k)_{k1,K} \in \mathbb{R}^{|A|\times|K|}$.

**Definition 2** (Residual network)**.** *The residual network associated to $x^k$ is the graph denoted by $G^k(x) = (V, A^k(x))$ and defined by the set of activated and unsaturated arcs $A^k(x)$. This set of arcs is defined as the union of the two following sets:*

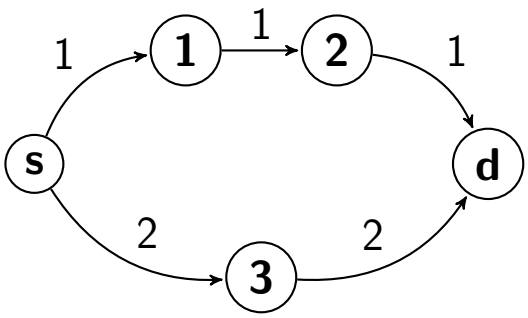
$$A^{k+}(x) = \left\{ (i,j) \in A : 0 < \sum_{k' \in K} x^{k'}_{ij} < C_{ij}, \quad x^k_{ji} = 0 \right\}$$
$$A^{k-}(x) = \left\{ (i,j) \in A : \qquad x^k_{ji} > 0 \right\}.$$

*Each arc of $G^k(x)$ is associated with a residual cost $c'^k$ defined as:*

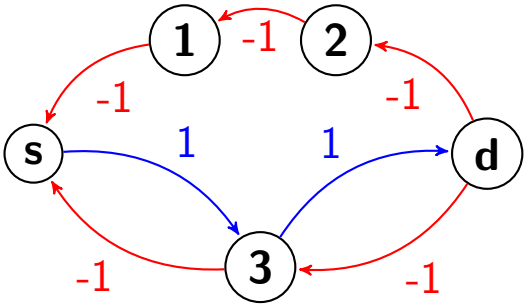
$$c'^k_{ij} = \begin{cases} \frac{1}{C_{ij}}, & \text{if } \ (i,j) \in A^{k+}, \\ -\frac{1}{C_{ij}}, & \text{if } \ (i,j) \in A^{k-}. \end{cases}$$



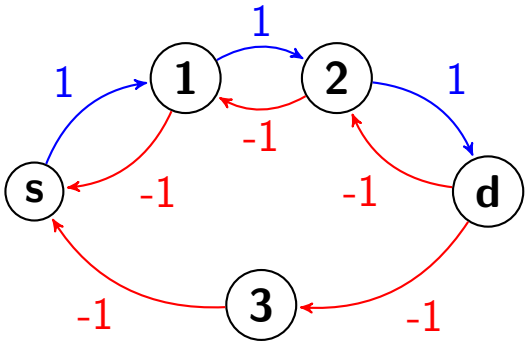

(a) Unfeasible flow.




(b) Feasible flow.




(c) The residual network of (4.3a).




(d) The residual network of (4.3b).

Figure 4.3: Example of an unfeasible (feasible) flow violating (satisfying) negative cycle conditions. All link capacities are equal to 2 with $\Phi = 3$ to be sent from $s$ to $d$.

**Theorem 12** (Residual network optimality ([4], chapter 9))**.** *A feasible flow $x^k$ of a demand $k$ is optimal if and only if the network $G(x^k)$ has no negative cycles. This condition is equivalent to:*

$$c'^k_{ij} + \Pi^k_i - \Pi^k_j \geq 0, \qquad \forall (i,j) \in G(x^k), \forall k \in K. \tag{4.5a}$$

To illustrate, let us consider the example in Figure 4.3. We suppose that the capacity of each link is equal to 2 and we have a demand with $\phi = 3$ to be sent from the node s to d. Figure 4.3 represents two solutions (4.3a) and (4.3b) and the associated residual networks (4.3c) and (4.3d), respectively. The solution in Figure (4.3a) is unfeasible because the associated residual network contains

a negative cycle: s-3-d-2-1-s (resp,(4.3b) is feasible because (4.3d) contains no negative cycle).

We can model the residual network associated with a demand $k$ by using binary variables $\mu_{ij}^{k+}$, $\mu_{ij}^{k-}$, and the additional variables $\nu_{ij}$ for each $(i,j) \in A$.

**Proposition 2.** *Let* $\mu \in \{0,1\}^{2\times|A|\times|K|}, \nu \in \{0,1\}^{|A|}$ *be a solution of:*

$$C_{ij} - \sum_{k' \in K} x_{ij}^{k'} \leq C_{ij}\nu_{ij}, \qquad \forall (i,j) \in A, \tag{4.6a}$$

$$\mu_{ij}^{k+} + \mu_{ij}^{k-} \leq y_{e(i,j)}, \qquad \forall (i,j) \in A, \tag{4.6b}$$

$$\mu_{ij}^{k-} \leq x_{ji}^{k} \qquad \forall (i,j) \in A, \tag{4.6c}$$

$$C_{ji}\mu_{ij}^{k-} \geq x_{ji}^{k} \qquad \forall (i,j) \in A, \tag{4.6d}$$

$$C_{ij}\mu_{ij}^{k+} \leq C_{ij} - \sum_{k' \in K} x_{ij}^{k'} \qquad \forall (i,j) \in A, \tag{4.6e}$$

$$\mu_{ij}^{k+} \geq y_e - \mu_{ij}^{k-} + \nu_{ij} - 1 \qquad \forall (i,j) \in A, \tag{4.6f}$$

$$\mu_{ij}^{k+}, \mu_{ij}^{k-} \in \{0,1\} \qquad \forall (i,j) \in A, \tag{4.6g}$$

$$\nu_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. \tag{4.6h}$$

1. *For each* $(i,j) \in A$, $\nu_{ij} = 1$ *if:* $\sum_{k \in K} x_{ij}^{k} < C_{ij}$.
2. *Arc* $(i,j) \in A^{k-}(x)$ *iff:* $\mu_{ij}^{k-} = 1$.
3. *Arc* $(i,j) \in A^{k+}(x)$ *iff:* $\mu_{ij}^{k+} = 1$.

*Proof.* We see that the set of constraints (4.6a)-(4.6h) is decomposable by arc. Hence, consider an arc $(i,j) \in A$.

1. is guaranteed by constraints (4.6a).
2. is guaranteed by constraints (4.6c) and (4.6d):
   - $(i,j) \notin A^{k-} \Rightarrow \quad x_{ji}^{k} = 0 \quad (4.6c) \quad \mu_{ij}^{k-} = 0.$

- $(i,j) \in A^{k-} \Rightarrow \quad x^k_{ji} > 0 \quad (4.6d) \quad \mu^{k-}_{ij} = 1.$

3. is guaranteed by constraints (4.6a)-(4.6f):

- $(i,j) \in A^{k+} \quad (4.6a), (4.6c) \quad \mu^{k-}_{ij} = 0 \text{ and } \nu_{ii} = 1 \quad (4.6f) \quad \mu^{k+}_{ij} = 1.$
- $(i,j) \notin A^{k+}$Definition 2 $\begin{cases} C_{ij} - \sum_{k \in K} x^k_{ij} = 0 \quad (4.6e) \quad \mu^{k+}_{ij} = 0, \\ \text{or} \\ x^k_{ji} > 0 \qquad\qquad (4.6c), (4.6f) \quad \mu^{k+}_{ij} = 0. \end{cases}$

Hence, for any solution to (4.6), arc $(i,j) \in A^{k+}(x)$ (resp. $(i,j) \in A^{k-}(x)$) if $\mu^{k+}_{ij} = 1$ (resp. $\mu^{k-}_{ij} = 1$). □

Using Theorem 12 and Proposition 2, the ETE-MPR problem can be reformulated with the residual network optimality conditions.

$$\begin{aligned}
\min \quad & \sum_{i \in V} P_i z_i + \sum_{(i,j) \in A} \sum_{k \in K} g_{ij} x^k_{ij} \\
\text{s.t.} \quad & (4.1b) - (4.1e), (4.6a) - (4.6h), \\
& \sum_{(i,j) \in \delta^+(i)} x^k_{ij} - \sum_{(i,j) \in \delta^-(i)} x^k_{ji} = b^k_i, && \forall i \in V, \forall k \in K, && (4.7a) \\
& \sum_{k \in K} x^k_{ij} \leq C_{ij} y_{e(i,j)}, && \forall (i,j) \in A, && (4.7b) \\
& \frac{1}{C_{ij}} + \pi^k_i - \pi^k_j \geq M(\mu^{k+}_{ij} - 1) && \forall (i,j) \in A, \forall k \in K && (4.7c) \\
& -\frac{1}{C_{ij}} + \pi^k_i - \pi^k_j \geq M(\mu^{k-}_{ij} - 1) && \forall (i,j) \in A, \forall k \in K && (4.7d) \\
& x^k_{ij} \geq 0, && \forall (i,j) \in A, \forall k \in K. && (4.7e)
\end{aligned}$$

Constraints (4.7c) and (4.7d) eliminate negative cycles in the residual network associated to each flow $x^k$, for all $k \in K$. Since these constraints represent

optimality condition (4.5a) associated to our problem.

### 4.3.5 Single path routing

Section 4.6 presents a comparison of the multi-path routing model with the existing single-path model. Thus, in this subsection, we describe the bi-level problem of minimizing energy and link utilization while using a single path protocol OSPF. The single path routing problem consists of finding a single shortest path for each demand $k \in K$ from its source node $o(k)$ to the destination node $d(k)$. This problem can be formulated as:

$$\min \quad \sum_{i \in V} P_i z_i + \sum_{(i,j) \in A} \sum_{k \in K} g_{ij} \phi^k x_{ij}^k$$

$$\text{s.t.} \quad (4.1b) - (4.1e),$$

$$\min \quad \sum_{k \in K} \sum_{(i,j) \in A} \frac{1}{C_{ij}} x_{ij}^k \tag{4.8a}$$

$$\text{s.t.} \sum_{(i,j) \in \delta^+(i)} x_{ij}^k - \sum_{(i,j) \in \delta^-(i)} x_{ji}^k = b_i^k, \qquad \forall i \in V, \forall k \in K, \tag{4.8b}$$

$$\sum_{k \in K} \phi^k x_{ij}^k \leq C_{ij} y_{e(i,j)}, \qquad \forall (i,j) \in A, \tag{4.8c}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall (i,j) \in A, \forall k \in K, \tag{4.8d}$$

where:

$$b_i^k = \begin{cases} 1, & \text{if } \ i = o(k), \\ -1, & \text{if } \ i = d(k), \\ 0, & \text{otherwise.} \end{cases}$$

$x^k \in \{0,1\}^{|A|}$ represents here the path for demand $k$. Then, the flow in each arc $(i,j) \in A$ will be given by $\phi^k x_{ij}^k$ in this single-path routing model.

The follower problem is now an integer linear program. Thus, the KKT opti-

mality conditions cannot be applied here. We propose to formulate the single path routing problem as a one level BILP program with flow elimination constraints as:

$$\min \quad \sum_{i \in V} P_i z_i + \sum_{(i,j) \in A} \sum_{k \in K} g_{ij} \phi^k x_{ij}^k$$

$$\text{s.t.} \quad (4.1b) - (4.1e),$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij}^k - \sum_{(i,j) \in \delta^-(i)} x_{ji}^k = b_i^k, \qquad \forall i \in V, \forall k \in K, \tag{4.9a}$$

$$\sum_{k \in K} \phi^k x_{ij}^k \leq C_{ij} y_{e(i,j)}, \qquad \forall (i,j) \in A, \tag{4.9b}$$

$$\sum_{k \in K} \sum_{(i,j) \in A} \frac{1}{C_{ij}} x_{ij}^k \leq C(P) + (|E(P)| - \sum_{e \in E(P)} y_e) M, \qquad \forall P \in \mathcal{P}, \tag{4.9c}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall (i,j) \in A, \forall k \in K. \tag{4.9d}$$

where, $\mathcal{P}$ is the set of all feasible solutions $P = (x^*, y^*, z^*)$, and $E(P)$ is the set of activated arcs on $P$ defined by $E(P) = \{e \in E : \ y^* = 1\}$. In (4.9c), $C(P)$ represents the total cost of $P$, and $M$ is a precomputed large number that can be replaced by $\sum_{k \in K} \Phi_k$. The single path routing problem is studied in [28] without capacity constraints, in this paper the bi-level problem is reformulated using the constraint (4.9c) which gave good results comparing the other proposed reformulations.

## 4.4 Alternative application: Minimising energy in Data Center Networks

Energy consumption has become a critical problem not only in telecommunication but also in data center networks. In the last years, one of the most fundamental service models is the Cloud Computing. Consequently, many companies as Microsoft, Google, and Amazon have deployed large data-centers to provide adequate computing resources in clouds. To provide robust computing capabilities, these data centers generate a large amount of energy waste due to inefficient use of physical resources, leading to higher expenses and environmental concerns [137].

In this application, we study the problem of energy management in data center networks (**DCNs**) using multi-path routing with minimum link capacity utilization.

We consider a data-center network modeled as a distributed computing system represented by the network $G = (V, E)$ where the set of nodes $V$ represents connected servers the network (switches and hosts) and $E$ is the set of communication network links. Similarly to the precedent application, we suppose that the energy management function and the deployed routing protocol operate independently by two different agents. On the one hand, to improve the energy efficiency, the energy management function of the ISP network tends to minimize the network energy consumption by putting on sleep unused network devices according to the actual traffic demands. On the other hand, the network operator should also ensure that the deployed routing protocol is able to handle all traffic demands within the available capacities while minimizing the capacity link utilization.

In the literature, the problem of minimizing energy consumption in data center networks was never considered as bi-level model. Our bi-level model is motivated by the existence of two uncooperative network operators with different objectives.

The two levels of the problem can be defined as:

**Upper level:** The data center provider minimizes the total energy consumption of the network by putting on sleep the unused devices according to the traffic demands.

**Lower level:** The network operator minimizes the total capacity utilization using a multi-path routing protocol.

The problem can be reformulated as a bi-level MILP problem, as follows:

$$\min \quad \sum_{i \in V} P_i z_i + \sum_{(i,j) \in A} \sum_{k \in K} g_{ij} x_{ij}^k \tag{4.10a}$$

$$\text{s.t.} \quad z_i \in \{0,1\}, \qquad \forall i \in V, \tag{4.10b}$$

$$y_e \geq z_i + z_j - 1, \qquad \forall e = \{i,j\} \in E, \tag{4.10c}$$

$$y_e \leq z_i, \qquad \forall e = \{i,j\} \in E, \tag{4.10d}$$

$$y_e \leq z_j, \qquad \forall e = \{i,j\} \in E, \tag{4.10e}$$

$$\min \quad \sum_{k \in K} \sum_{(i,j) \in A} \frac{1}{C_{ij}} x_{ij}^k \tag{4.10f}$$

$$\text{s.t.} \quad \sum_{(i,j) \in \delta^+(i)} x_{ij}^k - \sum_{(i,j) \in \delta^-(i)} x_{ji}^k = b_i^k, \qquad \forall i \in V, \forall k \in K, \tag{4.10g}$$

$$\sum_{k \in K} x_{ij}^k \leq C_{ij} y_{e(i,j)}, \qquad \forall (i,j) \in A, \tag{4.10h}$$

$$x_{ij}^k \geq 0, \qquad \forall (i,j) \in A, \forall k \in K. \tag{4.10i}$$

where:

$$b_i^k = \begin{cases} \phi^k, & \text{if } \ i = o(k), \\ -\phi^k, & \text{if } \ i = d(k), \\ 0, & \text{otherwise.} \end{cases}$$

The upper-level objective function (4.10a) aims to minimize the energy consumption of DCN devices, while the objective of the second level problem is to minimize the total link capacity utilization (4.10f). Constraints (4.10c)-(4.10e), (4.10g) and (4.10h) represent, respectively, the activation devices, flow conservation and capacity constraints.

Since the two presented applications have the same bi-level model in different networks, in the rest of this chapter we will consider only the first application (Energy management in the ISP backbone networks), and the proposed algorithms can solve both of them.

## 4.5 Agorithms

In Subsections 4.3.2, 4.3.3, and 4.3.4, we presented three different formulations for ETE-MPR the problem. Formulations (4.3) and (4.7) are compact ones, while formulation (4.4) may have an exponential number of constraints. In this section, first, we focus on presenting different exact methods to deal with the problem formulations (Subsections 4.5.1, 4.5.2, and 4.5.3). Then, we present how formulation (4.8) will be solved (Subsection 4.5.4).

### 4.5.1 Compact formulations

The first way to solve the problem is to feed the one level models based on KKT (4.3) and residual network optimality conditions (4.7) to an ILP solver with default parameters. Let **KKT** and **RN** denote the solution methods obtained, respectively.

### 4.5.2 Cutting plane algorithm

In this subsection, we discuss how to solve our problem with a cutting plane (**CP-FC**) algorithm based on the MILP formulation (4.4). This algorithm is described in the following:

1. Let $\mathcal{Q}$ be the first level problem minimizing the sum of the energy consumption and the arc utilization:

$$\begin{aligned} \min \quad & \sum_{i \in V} P_i z_i + \sum_{(i,j) \in A} \sum_{k \in K} g_{ij} x_{ij}^k \\ \text{s.t.} \quad & (4.1\text{b}) - (4.1\text{e}), (4.1\text{g}) - (4.1\text{i}). \end{aligned}$$

2. Solve $\mathcal{Q}$ and let $F^* = (z^*, y^*, x^*)$ be the solution obtained.

3. Let $N^*$ be the subnetwork defined by the set of activated routers $(z^*, y^*)$, and solve the second level problem (minimization of the link capacity utilization) in $N^*$. Let $x'$ be the flow obtained and $F' = (z^*, y^*, x')$.

4. If $C(F) = C(F')$, then the solution $(z^*, y^*, x^*)$ is optimal, stop. Otherwise, add the inequality (4.4b) associated with the solution $F^*$ as a new constraint to $\mathcal{Q}$ in order to cut the unfeasible solution $(z^*, y^*, x^*)$.

5. Go to step 2. The procedure described above is repeated until the optimal solution is found.

### 4.5.3 Branch-and-cut algorithm

The one-level formulation (4.4) can also be used to solve the ETE-MPR problem by a branch-and-cut (**B&C-FC**) algorithm.

The main idea of the algorithm used here is to solve the first level problem (without taking in consideration the minimization of the total traffic) by a branch-and-bound procedure and to add valid inequalities (4.4b) at each integral node of branch-and-bound tree violating the min-cost flow constraints. Once an integral node of the branch-and-bound tree is reached, i.e. a node where an integer solution has been found, if an inequality (4.4b) is violated it is added to the problem through callbacks. In the sequence, the algorithm continues the search for an integer optimal solution satisfying all flow constraints. The algorithm stops when there are no more nodes to evaluate, i.e., all the required missing flow constraints (4.4b) were generated.

The difference between this algorithm and the iterative cutting plane algorithm presented in Subsection 4.5.2 lies in solving a unique MILP while one MILP is solved from scratch at each iteration of the algorithm previous presented.

### 4.5.4 Single path routing

The single path routing is a generalization of the fixed charge network design problem with shortest path constraints (FCNDP-SCP) studied in [28]. Because of the capacity constraints added to the problem, the algorithms proposed in [28] are no longer applicable.

One of the ways to solve it is to use the iterative cutting plane algorithm presented in Subsection 4.5.2. The only difference is that we now solve the first level problem of (4.8), and we cut a path violating shortest path requirement in each iteration by adding valid inequalities (4.9c).

## 4.6 Numerical results

In this section, we present two different experiments. First, we compare in Subsection 4.6.2 the CPU time of the algorithms presented in subsections 4.5.1-4.5.3 to obtain optimal solutions. Then, in order to compare our multi-path bi-level model with the existing single-path model, numerical evaluations are performed to compare the improvement on values of optimal solutions in Subsection 4.6.3.

All algorithms are implemented in Julia 0.6.0, and the problems are solved using Gurobi 7.5.1 with default parameters and a time limit of $3600s$. Simulations were performed on an Intel(R)core TM i7-6500M CPU@2.50 2.60GHz computer with 8 GB of RAM.

### 4.6.1 Instances set

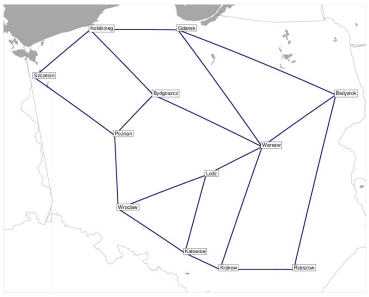
Our computational experiments have been carried out on different network topologies provided by [118] and depicted in Figure 4.4: **Abilene** (4.4a), **Polska** (4.4b), and **Geant** (4.4c) networks. The capacity of line cards and the energy consumption of each chassis are randomly chosen from the three cases in Table 4.1 (Table 1 [6]).

| Chassis Features | | | |
|---|---|---|---|
| Case | device | capacity | hourly power consumption |
| 1 | Gigabit-Eth 1 port | 2 Gbps | 7.3 W |
| 2 | Fast-Eth 12 ports | 2.4 Gbps | 18.6 W |
| 3 | Gigabit-Eth 4 ports | 8 Gbps | 31 W |
| 4 | SONET /SDH OC-48c | 5 Gbps | 41.4 W |

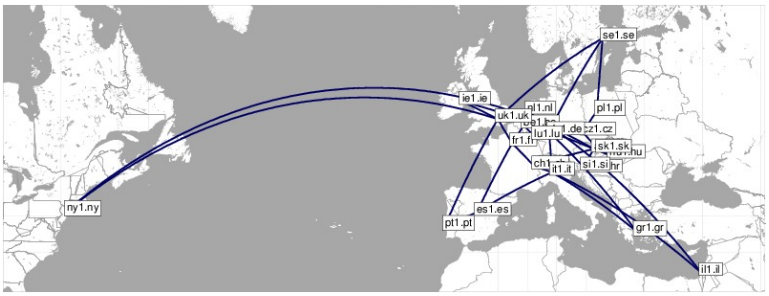
Table 4.1: Router chassis and cards

Five different instances are generated for each network topology. Let $|V_e|$ be the number of nodes selected to be origin or destination point, and $|K|$ is the number of demands. Network instances are summarized in Table 4.2.


(a) Abilene network



(b) Polska network




(c) Geant network

Figure 4.4: Examined network topologies: (a) Abilene (b) Polska (c) Geant [118].

| **Instances** | | | **1** | | **2** | | **3** | | **4** | | **5** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Network | $\|V\|$ | $\|E\|$ | $\|V_e\|$ | $\|K\|$ | $\|V_e\|$ | $\|K\|$ | $\|V_e\|$ | $\|K\|$ | $\|V_e\|$ | $\|K\|$ | $\|V_e\|$ | $\|K\|$ |
| **Abilene** | 12 | 15 | 4 | 5 | 5 | 10 | 6 | 20 | 7 | 30 | 8 | 40 |
| **Polska** | 12 | 18 | 4 | 5 | 5 | 10 | 6 | 20 | 7 | 30 | 8 | 40 |
| **Geant** | 22 | 36 | 4 | 5 | 5 | 10 | 6 | 20 | 7 | 30 | 8 | 40 |

Table 4.2: Network topologies used.

The demand between each origin-destination pair is randomly generated in $[100, 700]$ Mgbs.

### 4.6.2 Solution times

First, we compare the different algorithms presented in subsections 4.5.1-4.5.3 with respect to solution times. The obtained results are reported in Tables 4.3, 4.4 and 4.5. These tables present for each instance, the CPU time in seconds spent by the different algorithms to get the optimal solutions. The symbol in the

| Instances | KKT | RN | CP-FC | B&C-FC |
|---|---|---|---|---|
| 1 | 0.08 | 0.08 | 0.03 | 0.34 |
| 2 | 0.35 | 0.94 | 0.09 | 0.06 |
| 3 | 2.31 | 3.47 | 0.10 | 0.07 |
| 4 | 6.84 | 6.11 | 0.10 | 0.11 |
| 5 | 19.19 | 10.12 | 0.12 | 0.11 |
| Average | 5.76 | 4.14 | 0.09 | 0.14 |

Table 4.3: Comparison of CPU time on Abilene network.

tables of results "-" means that the algorithm was not able to find the optimal solution in the time limit of $3600s$ for the considered instance.

| Instances | KKT | RN | CP-FC | B&C-FC |
|---|---|---|---|---|
| 1 | 0.56 | 0.35 | 0.31 | 1.37 |
| 2 | 4.63 | 0.53 | 0.28 | 0.40 |
| 3 | 3600.02 | 4.03 | 0.82 | 0.67 |
| 4 | - | 185.96 | 4.67 | 2.08 |
| 5 | - | 3180.68 | 86.03 | 1.01 |
| Average | 1 201.74 | 674.31 | 18.42 | 1.11 |

Table 4.4: Comparison of CPU time on Geant network.

The results show that the iterative cutting plane and branch-and-cut algorithms are close in terms of CPU time and outperform significantly the compact formulations. Also, We can remark, comparing the different tables, that the Geant instance (Figure (4.4c)) is more difficult than the other instances in terms of CPU time. If we consider, for instance, the cutting plane algorithm, the CPU time average to find the optimal solutions for Geant instance is more than 200 times the CPU time of Abilene network and 100 times the CPU time of Polska network. Furthermore, for all instances, we observe that the first scenarios 1, 2 and 3 are easier than the last ones 4 and 5 according to the running time of all algorithms. These results can be explained by difference of the size and the number of demands in each scenario and instance.

| Instances | KKT | RN | CP-FC | B&C-FC |
|---|---|---|---|---|
| 1 | 0.17 | 0.16 | 0.04 | 0.23 |
| 2 | 1.18 | 1.80 | 0.21 | 0.16 |
| 3 | 0.67 | 2.33 | 0.22 | 0.12 |
| 4 | 11.83 | 24.26 | 0.22 | 0.18 |
| 5 | 1453.72 | 7.38 | 0.21 | 0.11 |
| Average | 293.51 | 7.18 | 0.18 | 0.16 |

Table 4.5: Comparison of CPU time on Polska network.

Furthermore, the residual network formulation is faster than the formulation based on KKT conditions.

### 4.6.3 Single path routing vs multi-path routing

We compared our multi-path model with the single-path approach (where each demand uses one path).

| | Abilene network | | Geant network | | Polska network | |
|---|---|---|---|---|---|---|
| Instances | multi-path | single path | multi-path | single path | multi-path | single path |
| 1 | 21531.7 | 21531.7 | 20856.8 | 20856.8 | 33900.4 | 33900.4 |
| 2 | 97076.9 | 97076.9 | 35949.8 | 35949.8 | 67535.8 | 67535.8 |
| 3 | 199014.3 | 199512.3 | 118786.9 | 132426.9 | 104429.8 | 107261.8 |
| 4 | 200820.3 | 204881.3 | 220255.3 | 223851.4 | 176674.7 | 181298.7 |
| 5 | 220454.3 | 224991.3 | 200819.3 | 203373.3 | 290679.3 | 291752.3 |
| Average | 147779.5 | 149598.7 | 119333.62 | 123291.64 | 134644 | 136349.8 |

Table 4.6: Comparison of the first level solution of the multi-path and the single path approaches.

Tables 4.6 and 4.7 summarize the obtained results. For each instance, we compare the first and the second level optimal solution obtained under the two settings: single and multi-path routing. We can easily remark that using the multi-path leads to lower network energy consumption (the first level objective function) for 9/15 instances. This is because in the single path approach we need to activate more routers and links in the case of insufficient link capacity,

| | Abilene network | | Geant network | | Polska network | |
|---|---|---|---|---|---|---|
| Instances | multi-path | single path | multi-path | single path | multi-path | single path |
| 1 | 4908 | 4908 | 6423 | 6423 | 7848 | 7848 |
| 2 | 21791 | 21791 | 13018 | 13018 | 14410 | 14410 |
| 3 | 39572 | 39771 | 31941 | 32396 | 26346 | 26346 |
| 4 | 40642 | 42176 | 49948 | 51548 | 36688 | 36688 |
| 5 | 44003 | 45892 | 50105 | 50450 | 56448 | 56459 |
| Average | 30183.2 | 30907.6 | 30287 | 30767 | 28348 | 28350.2 |

Table 4.7: Comparison of the second level solution of the multi-path and the single path approaches.

| | Abilene network | | Geant network | | Polska network | |
|---|---|---|---|---|---|---|
| Instances | First obj | second obj | First obj | second obj | First obj | second obj |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0.24 | 0.50 | 10.30 | 1.40 | 2.64 | 0 |
| 4 | 1.98 | 3.64 | 1.60 | 3.10 | 2.55 | 0 |
| 5 | 2.02 | 4.12 | 1.26 | 0.68 | 0.37 | 0.02 |
| Average | 0.85 | 1.65 | 2.63 | 1.04 | 1.11 | 0.004 |

Table 4.8: Percentage of the reduction on the first and the second level objectives using the multi-path model.

unlike the multi-path where the flow can be distributed. In addition, and for the same reason, in 7/15 instances, the total traffic (the second level problem) is reduced. To conclude, Table 4.8 reports the percentage of the reduction of energy consumption (First level objective) and capacity link utilization (Second level objective) when we use the multi-path approach. It is clear that, in the worst case, the two models give the same solution. Lastly, the quantity of the energy and the capacity of links saved even if the average the obtained gaps are small. Considering for example, the third instance of Geant network, the percentage of the energy reduced is small (10.30%), but, it represents more than 13000 (unity of energy).

### 4.6.4 Bi-level vs bi-objective model

| | Abilene network | | Geant network | | Polska network | |
|---|---|---|---|---|---|---|
| Instances | First obj | second obj | First obj | second obj | First obj | second obj |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 498 | 199 | 13640 | 455 | 2832 | 0 |
| 4 | 4061 | 1534 | 3596.1 | 1600 | 4624 | 0 |
| 5 | 4537 | 1889 | 2554 | 345 | 1073 | 11 |
| Average | 1819.2 | 724.4 | 3958.02 | 480 | 1705.8 | 2.2 |

Table 4.9: Table of the differnce between the first and the second level solution

Finally, we compare the difference between our bi-level model and the classical multi-objective approach. This latter is solved with epsilon constraint method to get the Pareto optimal front [65]. Both approaches here are solved with CPLEX solver. In Table 4.10, we compare the solution time of the two approaches, we can remark that the bi-level model is more efficient in terms of CPU time.

| | **Comparison of the first level objective function value** | | | | | |
|---|---|---|---|---|---|---|
| | Abilene network | | Geant network | | Polska network | |
| Instances | multi-objective | bi-level | multi-objective | bi-level | multi-objective | bi-level |
| 1 | 0.91 | 0.16 | 0.14 | 0.09 | 120.79 | 0.05 |
| 2 | 0.06 | 0.07 | 0.11 | 0.11 | 222.23 | 0.50 |
| 3 | 83.04 | 0.19 | 23.32 | 1.84 | 323.16 | 0.35 |
| 4 | 16.09 | 0.17 | 378.90 | 4.84 | 347.45 | 0.49 |
| 5 | 24.02 | 0.21 | 18.11 | 1.62 | 99.32 | 0.40 |
| Average | 24.82 | 0.16 | 84.12 | 1.70 | 222.59 | 0.36 |

Table 4.10: CPU time in seconds of the bi-level and multi-objective models.

# Chapter 5

# Maximum influence in Signed social networks

In this chapter, we study the problem of maximum influence in signed social networks. For the best of our knowledge, this work is the first one to study this problem on signed networks and to model it as a bi-level programming formulation. As done in the previous chapters, we reformulate the problem as one-level MILP models using different optimality conditions of the second level problem. These new formulations are strengthened by adding a set of valid inequalities. Computational experiments are performed using random instances to compare the different proposed formulations.

## 5.1 Introduction

In the last years, the number of users of social networks such as Facebook, YouTube, and Twitter rapidly increased (about 2.4 Billion users for Facebook, 1.9 billion for YouTube, and 330 million users of Twitter). In these social networks, people share and receive information, advertisements, and ideas from friends or

subscribers in "word-of-mouth" form of communication. From the users perspective, this provides to the users new and comfortable channels for exchanging information and expressing views and opinions [85; 138]. From the marketing perspective, this allows to the social media to penetrate in all aspects of everyday lives. In this context, the study of influence propagation through a social network gained importance when deciding whether or not to adopt an innovation, such as a political idea, a new product, or medical and technological innovations. In the literature, works studying the influence and effects of "word-of-mouth" in the promotion of new products in social networks are motivated by applications like the spreading of ideas or innovations in a network and viral marketing of products [31; 54; 67; 68]. Definitely, influence maximization has become a relevant problem on social networks [143].

The Maximum influence problem in social networks consists of selecting a subset of seeds (users of the network) to spreed information or ideas in order to maximize the spread of influence in the network. This problem has been treated in the literature under different assumptions. From a historical point of view, Domingos and Richardson [54; 124] were the first ones to propose a probabilistic model for this problem. In 2003, Kempe, Kleinberg, and Tardos [85] formulated for the first time the problem as a discrete optimization (MIP) problem, which they proved to be $\mathcal{NP}$-complete. After that, several papers were devoted to studying different versions of this problem [34; 35; 57; 74; 88; 128]. In particular, independent cascade [86; 87], linear threshold [74; 75], time-aware diffusion [33; 103], and many other diffusion models were presented. To solve the problem, in the works cited above and in other papers published later [67; 75; 87; 103], heuristic algorithms were proposed based on the different propagation models of the problem.

In this chapter, we are interested in signed social networks which contain both

positive and negative relationships (e.g., foe or distrust) between users. Influence maximization in signed social networks is still a challenging problem that has not been explicitly modeled as a signed graph optimization problem. In the literature, the problem studied consists in selecting $k$ users to be activated in order to spread the information in the network [100]. The objective of the problem is to maximize the expected spread of influence under certain predefined propagation models. The two classical propagation models used in the literature are probabilistic ones: Linear Threshold and Independent Cascade models. Based on the algorithms proposed for the same problem in unsigned networks, generalized models and approaches are proposed to solve threshold-based maximum influence in social network problem [7; 101; 102; 126; 139]. In these works, the different relationships on the social network are defined by modeling the problem in signed graphs or by using positive and negative weights for the links of the network.

In the present work, influence maximization in signed social networks is explicitly formulated as a signed graph optimization problem. The version of the problem studied here assumes two states of the information to be diffused (for example, a good or a bad opinion about a product). We study how to strategically select a subset of individuals (called here *seeds*) in these networks in order to diffuse the information in its original state to other users of the network. All assumptions made in this version of the problem take into consideration the signed relations between individuals. For example, information diffused through a negative link is inverted while information diffused through a positive link keeps its state.

In the first place, we formulate the problem as a linear bi-level problem. The first level problem involves binary variables to define the set of selected seeds and also the state of the information arriving to each user. The second level problem is a linear problem with continuous variables used to define the paths between the

seeds and the rest of the network. Then, as in previous chapters, reformulations and optimality conditions will be used to rewrite the problem as MILP models. For the problem treated in this chapter, we also describe the optimal solution for a special class of signed graphs, namely, for balanced signed graphs.

## 5.2 Problem description

We consider a social network, which can be modeled as an undirected signed graph $G = (V, E, s)$. The set of nodes $V$ represents the individuals of the considered network. The set of edges $E$ represents polarized relationships among individuals, such as foe $\times$ ally, trust $\times$ distrust. Thus, each edge $\{i, j\} \in E$ in this graph is associated with a sign $s_{ij} \in \{+, -\}$ where $s_{ij} = +$ stands for ally or trust while $s_{ij} = -$ for foe or distrust, according to the type of relationship between individuals $i$ and $j$. An edge $\{i, j\}$ is called positive (respectively, negative) if $s_{ij} = +$ (respectively, $s_{ij} = -$). Let us define $E^+$ ($E^-$) the set of positive (negative) edges in the graph, which implies $E = E^+ \cup E^-$.

We suppose the existence of an information to be spread in the network which can take two opposite states in $\{I_0, I_1\}$. Ideally, the owner of the information wants it to be spread at state $I_0$ to each node (individual) in the graph (social network). The information can be sent at state $I_0$, from the owner to a node $i \in V$ at a sending cost $f_i$; in this case $i$ is called a *seed*. The set of seed nodes in charge of diffusing the initial information to the non-seed nodes in the networks.

A value $d_{ij}$ is associated with each edge $\{i, j\} \in E$. Assuming $i$ (respectively, $j$) receives the information at a time $t_0$, then $j$ (respectively, $i$) receives the information at time $t_0 + d_{ij}$ through edge $\{i, j\}$. A negative edge $\{i, j\} \in E^-$ means the information inverts as it flows on the edge, while it keeps its state whenever $\{i, j\} \in E^+$. Furthermore, a penalization $C_i$ is defined for each node $i$

receiving the information at state $I_1$, i.e. the reverse of the one initially sent by the owner.

Maximum influence problems amount to select a subset of nodes $S \subseteq V$ in the network (seeds) to diffuse a given information to each node $i \in V \setminus S$ [34; 85]. The version of the problem studied here assumes that, once the set of seeds $S$ is defined, the information retained by each non-seed node $i \in V \setminus S$ is the first information arriving to this node, i.e., the information retained arrives through a shortest path linking a seed $j \in S$ to $i$. This assumption is motivated by many psychology and marketing researches which prove the effect of the first impression on decision making [63; 91]; what is known as "Halo effect" for positive impressions and "Horn effect" for negative ones.

The Maximum Influence (Max-Inf) problem treated in this work looks for a set of seeds $S \subseteq V$ that minimizes the associated sending and penalization costs for all nodes in $V$.

As we have mentioned in the introduction of this chapter, the Max-Inf problem can be naturally considered as a bi-level optimization problem. In the upper level problem, the leader plays the role of the information owner who wants to minimize the sum of sending information costs and negative influence penalties. The second level problem represents an influence protocol where each user of the network (follower) is supposed to adopt the information that first reaches her from the chosen seeds. Hence the influence protocol relies on a shortest path problem. The bi-level problem treated in this chapter relates with the two other problems treated in previous chapters. From one hand, the second level problem is a shortest path problem like Chapter 3. From the other hand, the second level problem defines a predefined protocol as in Chapter 4. One important difference from the previous bi-level problems, concerns the set of decision variables of the leader which activates nodes instead of links. Our bi-level formulation will be

presented in the next section.

## 5.3 Mathematical formulations

In this section, we will introduce the mathematical formulations for the Max-Inf problem. We start by a bi-level integer programming model in Subsection 5.3.1. Then, the model will be transformed into three one level MILP formulations in Subsections 5.3.2, 5.3.3, and 5.3.4. Finally, a path based formulation is presented in Subsection 5.3.5. In the following, let $A = \{(i,j),(j,i) : \{i,j\} \in E\}$ be the set of arcs obtained by bi-directing each edge in $E$. We extended the distance vector $d$ and we define $d_{ij} = d_{ji}$ for each $\{i,j\} \in E$. Also, the set of negative (resp, positive) arcs on the graph is denoted by $A^- = \{(i,j) \in A : \ s_{ij} = -\}$ (resp, $A^+ = \{(i,j) \in A : \ s_{ij} = +\}$).

### 5.3.1 Bi-level programming formulation

To formulate the problem, let us consider binary variables $z_i^k$, for each $i,k \in V$, indicating whether node $i$ is chosen as a seed node and whether node $i$ influences node $k$. Precisely:

$$z_i^k = \begin{cases} 1, & \text{if } i \neq k \text{ and i is the seed of the information arriving to } k, \\ 1, & \text{if } i = k \text{ and i is a seed,} \\ 0, & \text{otherwise.} \end{cases}$$

We use variables $y_{ij}^k$, $(i,j) \in A$ and $k \in V$, to track the path used to send the information from a seed to a non-seed destination node $k$,

$$y_{ij}^k = \begin{cases} 1, & \text{if the arc } (i,j) \in A \text{ is in the path arriving to node k,} \\ 0, & \text{otherwise.} \end{cases}$$

Also, a binary variable $\pi_k$, $k \in V$, denotes whether the information arrives to node $k$ at state $I_0$ (as sent by the owner) or at state $I_1$ (inverted). Clearly, according to the problem definition, the value of $\pi_k$ depends on the number of negative arcs in the shortest path from a seed node to $k$. These decision variables can be defined as:

$$\pi_k = \begin{cases} 1, & \text{if the information arrives to node } k \text{ through a negative path,} \\ 0, & \text{if the information arrives to node } k \text{ through a positive path,} \end{cases}$$

where a path is said negative (resp. positive) if it contains an odd (resp. even) number of negative links. Then, $\pi_k = 1$ (resp. $\pi_k = 0$) if the information arrives to node $k$ at state $I_1$ (resp. $I_0$).

Next, we discuss the set of constraints used to define variables $\pi_k$. As we mentioned above, the variable $\pi_k$, with $k \in V$, indicates if the information is retained by node $k$ either at the original state ($I_0$) or inverted ($I_1$). Then, $\pi_k$ can be defined based on both, the sign of the last arc $(i,k)$ in the shortest path arriving to $k$ (i.e., with $y_{ik}^k = 1$) and on the state of the information retained by

node $i$:

$$\pi_k \geq y_{ik}^k - \pi_i, \qquad \forall (i,k) \in A^-, \tag{5.1a}$$

$$\pi_k \leq 2 - y_{ik}^k - \pi_i, \qquad \forall (i,k) \in A^-, \tag{5.1b}$$

$$\pi_k \geq y_{ik}^k + \pi_i - 1, \qquad \forall (i,k) \in A^+, \tag{5.1c}$$

$$\pi_k \leq 1 - y_{ik}^k + \pi_i, \qquad \forall (i,k) \in A^+, \tag{5.1d}$$

$$\pi_k \leq 1 - z_k^k, \qquad \forall k \in V. \tag{5.1e}$$

Constraints (5.1a) and (5.1b) force the values of $\pi_k$ and $\pi_i$ to be inverted whenever the arc used is a negative one, i.e., $(i,j) \in A^-$. Likewise, constraints (5.1c) and (5.1d) impose the values of these variables to be the same whenever a positive arc $(i,j) \in A^+$ is used. That means, node $j$ receives the same information as node $i$ if the arc $(i,j)$ is positive and the reversed information otherwise. Finally, constraint (5.1e) imposes that $\pi_k$ to be equal to 0 for all seed nodes.

The correct value for a variable $\pi_k$ can also be found by calculating the number of negative links in the path defined by variables $y_{ij}^k$. Then, a variable $\pi_k$ takes either the value 0 if the shortest path to $k$ contains an even number of negative links (positive path) or the value 1 otherwise (negative path). Thus, these constraints can be given as:

$$\left\lceil \frac{\sum_{(i,j)\in A^-} y_{ij}^k}{2} \right\rceil - \frac{\sum_{(i,j)\in A^-} y_{ij}^k}{2} \leq \pi_k \leq \left\lceil \frac{\sum_{(i,j)\in A^-} y_{ij}^k}{2} \right\rceil - \frac{\sum_{(i,j)\in A^-} y_{ij}^k}{2} + \frac{1}{2}, \quad \forall k \in V,$$

which is equivalent to:

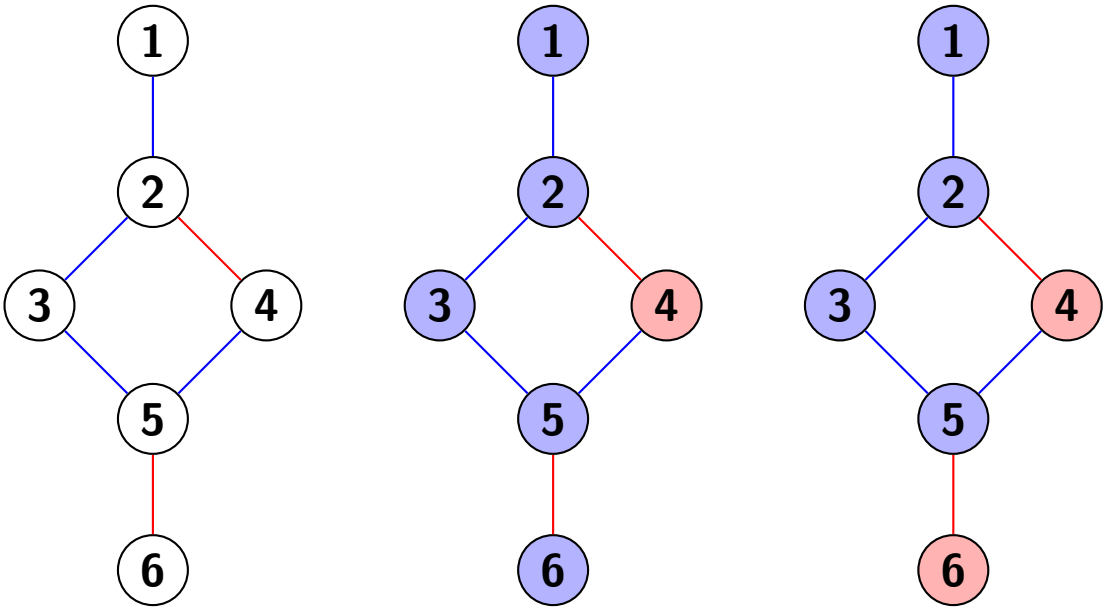
$$\frac{\sum_{(i,j)\in A^-} y_{ij}^k}{2} \leq x_k \leq \frac{\sum_{(i,j)\in A^-} y_{ij}^k + 1}{2}, \qquad \forall k \in V, \tag{5.2a}$$

$$x_k - \frac{\sum_{(i,j)\in A^-} y_{ij}^k}{2} \leq \pi_k \leq x_k - \frac{\sum_{(i,j)\in A^-} y_{ij}^k}{2} + \frac{1}{2}, \qquad \forall k \in V, \tag{5.2b}$$

$$x_i \in \mathbb{N}, \qquad \forall i \in V. \tag{5.2c}$$

Here, the value of each variable $\pi_k$ is defined according (only) to the number of negative links in the shortest path from a seed node to $k$. Notice that it is not related with the values of other variables $\pi_l$, $l \neq k$. As a consequence, in case of multiple shortest paths, the paths chosen can be incompatible, i.e, a node $k$ may receive the information with two opposite signs from different shortest paths represented by variables $y_{ik}^k$ and $y_{jk}^k$, $i \neq j$. This case is illustrated by Figure 5.1. Consider the signed graph depicted in Figure 5.1a: an arc with blue color represents a positive arc while a red arc represents a negative one. All link distances, costs, penalties are equal to 1. If we consider only constraints (5.2a)-(5.2c), the obtained solution is given by Figure 5.1b, where a blue node means the node receives information at state $I_0$ ($\pi_k = 0$) while a red node receives the information at state $I_1$ ($\pi_k = 1$). Node 1 is the only seed selected. Node 6 is influenced by node 1 receiving information at state $I_0$ through the path $(1-2-4-5-6)$. According to this shortest path, node 5 must receive information at state $I_1$. However, the solution says that node 5 receives information at state $I_0$ through the path $(1-2-3-5)$. Then, this solution is in fact infeasible for the Max-Inf problem. Since node 6 has to receive the information from 5, the path of 5 must be a part of the path of 6. After adding this condition, the obtained solution is presented in Figure 5.1c which is indeed compatible with the definition

of the Max-Inf problem.




(a) Signed graph. (b) Infeasible solution. (c) Feasible solution.

Figure 5.1: Effect of multiple shortest paths when constraints (5.2a)-(5.2c) are used to define variables $\pi_k$.

To avoid the problem in the presence of multiple shortest paths, we introduce the set of constraints:

$$\sum_{i' \in V \setminus \{i\}} \sum_{k \in V} y^k_{i'j} \leq M(1 - y^j_{ij}), \qquad \forall (i,j) \in A, \tag{5.3}$$

which forces the shortest path arriving to node $j$ to be a sub-path of each shortest path arriving to a node $k \in V$ passing through $j$.

To sum up, variables $\pi_k$, for each $k \in V$, can be defined based either on the state of the information received by the node sending the information to $k$, or by the number of negative arcs in the shortest path arriving to $k$. To distinguish the two different ways of defining these variables, let us introduce the two sets

$\Lambda_1$ and $\Lambda_2$ defined as:

$$\Lambda_1 = \{\pi \in \{0,1\}^n \ : (5.1a) - (5.1e)\},$$

$$\Lambda_2 = \{\pi \in \{0,1\}^n \, , x \in \mathbb{Z}^n \ : (5.2a) - (5.2c), (5.3)\}.$$

The bi-level model, presented in the following, selects at the first level the set of seed nodes which will receive the information directly from the owner. The second level computes the shortest paths connecting each node to a selected seed.

$$\min \quad \sum_{k \in V} f_k z_k^k + \sum_{k \in V} C_k \pi_k \tag{5.4a}$$

$$\text{s.t.} \quad z_i^k \in \{0,1\}\,, \qquad \forall i,k \in V, \tag{5.4b}$$

$$z_k^i \leq z_k^k, \qquad \forall i \neq k \in V, \tag{5.4c}$$

$$\pi_k \in \Lambda_1 \text{ or } \pi_k \in \Lambda_2, \qquad \forall k \in V, \tag{5.4d}$$

$$\min \quad \sum_{k \in K} \sum_{(i,j) \in A} d_{ij} y_{ij}^k \tag{5.4e}$$

$$\text{s.t.} \sum_{j:(k,j) \in A} y_{kj}^k - \sum_{j:(j,k) \in A} y_{jk}^k = z_k^k - 1, \qquad \forall k \in V, \tag{5.4f}$$

$$\sum_{j:(i,j) \in A} y_{ij}^k - \sum_{j:(j,i) \in A} y_{ji}^k = z_i^k, \qquad \forall i \neq k \in V, \tag{5.4g}$$

$$y_{ij}^k \leq 1 - z_k^k, \qquad \forall (i,j) \in A, \forall k \in V, \tag{5.4h}$$

$$y_{ij}^k \in \{0,1\}, \qquad \forall (i,j) \in A, \forall k \in V. \tag{5.4i}$$

The objective function of the leader (5.4a) minimizes the sum of sending costs associated with the selected seeds and penalty costs associated with non-seed nodes. Constraints (5.4c) allow node $i$ to receive information through a path beginning at node $k$ only if $k$ is selected as a seed node. The second level objective function (5.4e) minimizes the length of the path arriving to each node

$k \in V$ from all seed nodes in the graph. The set of constraints in the second level defines the set of paths used to transmit the information to all non-seed nodes. Constraints (5.4f)-(5.4h) are the flow conservation constraints. Constraints (5.4f) and (5.4g) force each non-seed node to receive an information from exactly one seed. Constraints (5.4h) force variables $y^k \in \{0,1\}^{|A|}$ to be equal to 0 whenever node $k$ is a seed (i.e., $z_k^k = 1$) avoiding shortest paths ending in seed nodes.

The second level problem is a shortest path problem between all pairs of nodes. We can also remark that the set of constraints of this problem (5.4f)-(5.4i) are represented by a totally unimodular matrix. Thus constraints (5.4i) can be replaced by:

$$0 \leq y_{ij}^k \leq 1, \qquad \forall (i,j) \in A, \quad \forall k \in V,$$

which can be reduced to a set of non-negativity constraints on $y$, from the presence of constraints (5.4h) in the second level. Hence, the second level problem becomes a continuous linear program.

### 5.3.2 One-level formulation

Similar to the problems studied in Chapters 3 and 4, the bi-level problem presented in the previous section can be reformulated as a one-level integer programming problem by replacing the second level problem by the associated optimality conditions: KKT, Bellman optimality conditions, and shortest path constraints.

Let us start by the fundamental complementary slackness conditions. The

KKT optimality conditions of the second level problem are given by:

$$\begin{cases} d_{ij} + \lambda_i^k - \lambda_j^k + \gamma_{ij}^k - \mu_{ij}^k = 0, & \forall (i,j) \in A, \forall k \in V, \\ y_{ij}^k - 1 + z_i^k = 0, & \forall (i,j) \in A, \forall k \in V, \\ \mu_{ij}^k y_{ij}^k = 0, & \forall (i,j) \in A, \forall k \in V, \\ \gamma_{ij}^k, \mu_{ij}^k \geq 0, & \forall k \in V, \forall (i,j) \in A, \\ \lambda_i^k \in \mathbb{R}, & \forall i, k \in V. \end{cases}$$

This set of constraints contains nonlinear inequalities. Using the Big-M linearization technique, we get the following one-level MILP formulation:

$$\begin{aligned} \min \quad & \sum_{k \in V} f_k z_k^k + \sum_{k \in V} C_k \pi_k \\ \text{s.t.} \quad & (5.4b) - (5.4d), \\ & \sum_{j:(k,j) \in A} y_{kj}^k - \sum_{j:(j,k) \in A} y_{jk}^k = z_k^k - 1, && \forall k \in V, && (5.5a) \\ & \sum_{j:(i,j) \in A} y_{ij}^k - \sum_{j:(j,i) \in A} y_{ji}^k = z_i^k, && \forall i \neq k \in V, && (5.5b) \\ & y_{ij}^k \leq 1 - z_k^k, && \forall (i,j) \in A, \forall k \in V, && (5.5c) \\ & d_{ij} + \lambda_i^k - \lambda_j^k + \gamma_{ij}^k \leq M(1 - y_{ij}^k), && \forall (i,j) \in A, \forall k \in V, && (5.5d) \\ & d_{ij} + \lambda_i^k - \lambda_j^k + \gamma_{ij}^k \geq 0, && \forall (i,j) \in A, \forall k \in V, && (5.5e) \\ & \gamma_{ij}^k \leq M(y_{ij}^k + z_k^k), && \forall (i,j) \in A, \forall k \in V, && (5.5f) \\ & \lambda_i^k \in \mathbb{R}, && \forall i, k \in V, && (5.5g) \\ & \gamma_{ij}^k,\ y_{ij}^k \geq 0, && \forall (i,j) \in A, \forall k \in V, && (5.5h) \end{aligned}$$

where $M$ is a large number.

### 5.3.3 Bellman based BILP formulation

As we mentioned before, the lower level problem is a shortest path problem. Hence, the problem can be reformulated by using Bellmans's optimality conditions presented in Chapter 2 and used in [28].

$$\min \quad \sum_{k\in V} f_k z_k^k + \sum_{k\in V} C_k \pi_k$$

$$\text{s.t.} \quad \text{(5.4b)} - \text{(5.4d)},$$

$$\sum_{j:(k,j)\in A} y_{kj}^k - \sum_{j:(j,k)\in A} y_{jk}^k = z_k^k - 1, \qquad \forall k \in V, \tag{5.6a}$$

$$\sum_{j:(i,j)\in A} y_{ij}^k - \sum_{j:(j,i)\in A} y_{ji}^k = z_i^k, \qquad \forall i \neq k \in V, \tag{5.6b}$$

$$y_{ij}^k \leq 1 - z_k^k, \qquad \forall (i,j) \in A, \forall k \in V, \tag{5.6c}$$

$$\Pi_i^k - \Pi_j^k \leq d_{ij} - 2d_{ij} y_{ji}^k, \qquad \forall (i,j) \in A, \forall k \in V, \tag{5.6d}$$

$$\Pi_k^k = 0, \qquad \forall k \in V, \tag{5.6e}$$

$$\Pi_i^k \geq 0, \qquad \forall i, k \in V, \tag{5.6f}$$

$$y_{ij}^k \geq 0, \qquad \forall (i,j) \in A, \forall k \in V. \tag{5.6g}$$

Non-negative variables $\Pi_i^k$ represent the shortest path distance between nodes $i$ and $k$. Then, for $k \in V$, $\Pi_k^k$ is set to be equal to zero in constraints (5.6e). Constraints (5.6d) are a lifted version of Bellman's optimality conditions, which guarantee the shortest path requirements. These constraints force $y_{ij}^k$ to be equal to 1 whenever $(i,j)$ is in the shortest path to $k$.

### 5.3.4 Shortest path formulations

Based on the definition of the problem, we can replace the second level problem by constraints eliminating infeasible paths violating the shortest path requirements.

Then, the problem can be reformulated as:

$$\min \quad \sum_{k \in V} f_k z_k^k + \sum_{k \in V} C_k \pi_k$$

$$\text{s.t.} \quad (5.4b) - (5.4d),$$

$$\sum_{j:(k,j) \in A} y_{kj}^k - \sum_{j:(j,k) \in A} y_{jk}^k = z_k^k - 1, \qquad \forall k \in V, \tag{5.7a}$$

$$\sum_{j:(i,j) \in A} y_{ij}^k - \sum_{j:(j,i) \in A} y_{ji}^k = z_i^k, \qquad \forall i \neq k \in V, \tag{5.7b}$$

$$y_{ij}^k \leq 1 - z_k^k, \qquad \forall (i,j) \in A, \forall k \in V, \tag{5.7c}$$

$$\sum_{(i,j) \in A} d_{ij} y_{ij}^k \leq D_h^k + M(1 - z_h^h), \qquad \forall k \neq h \in V, \tag{5.7d}$$

$$y_{ij}^k \geq 0, \qquad \forall (i,j) \in A, \forall k \in V, \tag{5.7e}$$

where, $D_h^k$ is the length of the shortest path from $h$ to $k$, which can be computed on $O(|V|^2.|E|)$.

We can write the set of shortest path constraints (5.7d) in a more compact form without using a Big-M constant. The Max-Inf problem can be reformulated by an alternative shortest path formulation as following:

$$\min \quad \sum_{k\in V} f_k z_k^k + \sum_{k\in V} C_k \pi_k$$

$$\text{s.t.} \quad (5.4b) - (5.4d),$$

$$\sum_{j:(k,j)\in A} y_{kj}^k - \sum_{j:(j,k)\in A} y_{jk}^k = z_k^k - 1, \qquad \forall k \in V, \quad (5.8a)$$

$$\sum_{j:(i,j)\in A} y_{ij}^k - \sum_{j:(j,i)\in A} y_{ji}^k = z_i^k, \qquad \forall i \neq k \in V, \quad (5.8b)$$

$$y_{ij}^k \leq 1 - z_k^k, \qquad \forall (i,j) \in A, \forall k \in V, \quad (5.8c)$$

$$\sum_{(h,j)\in A} y_{hj}^k \leq 1 - z_i^i, \qquad \forall i \neq h \neq k \in V, \text{ if: } D_i^k < D_h^k \quad (5.8d)$$

$$y_{ij}^k \geq 0, \qquad \forall (i,j) \in A, \forall k \in V, \quad (5.8e)$$

where, $D_i^k$ is the length of the shortest path from $i$ to $k$.

### 5.3.5 Path-based formulation

The problem can be also formulated by using variables defined on a set of paths. Let $P$ represents the set of all shortest paths $p$ between all pairs of nodes in the graph:

$$P = \{p : h \longrightarrow k, \text{ such that: } \text{length}(p) = D_h^k, \quad \forall h \neq k \in V\}.$$

The source and the destination of each path $p$ are denoted by $s(p)$ and $t(p)$, respectively. For a given node $k$, the set $P(k) \subset P$ contains all shortest paths arriving to the node $k$, i.e., $P(k) = \{p \in P : \ t(p) = k\}$. We also denote by $S_p$ the sign of path $p$: $S_p = 1$ if $p$ is a negative path; $S_p = 0$ if $p$ is a positive path. For the path-based formulation we introduce the following binary variables.

A variable $z_i$ indicates if node $i$ is selected as a seed, then, for all nodes $i \in V$

of the graph, we define:

$$z_i = \begin{cases} 1, & \text{if } i \text{ is a seed node,} \\ 0, & \text{otherwise.} \end{cases}$$

We define also a set of variables $Y_p$, for all $p \in P$, as:

$$Y_p = \begin{cases} 1, & \text{if the path } p \in P \text{ is in the solution,} \\ 0, & \text{otherwise.} \end{cases}$$

According to this definition, $Y_p = 1$ means that the destination node $t(p)$ of $p$ is influenced by the node $s(p)$ through the path $p$. Otherwise, the path $p$ does not belong to the solution.

The path based one-level model for the Min-Inf problem is as follows:

$$\begin{aligned}
\min \quad & \sum_{i \in V} f_i z_i + \sum_{p \in P} C_{t(p)} S_p Y_p \\
& \sum_{p \in P(k)} Y_p = 1 - z_k, && \forall k \in V, && \text{(5.9a)} \\
& Y_p \leq z_{s(p)}, && \forall p \in P, && \text{(5.9b)} \\
& Y_p \leq 1 - z_i, && \forall p : h \longrightarrow k,\ \forall i \in V : D_i^k < D_h^k, && \text{(5.9c)} \\
& Y_{p'} \geq Y_p, && \forall p \in P,\ \forall p' \subset p : s(p') = s(p), && \text{(5.9d)} \\
& z_i \in \{0, 1\}, && \forall i \in V, && \text{(5.9e)} \\
& Y_p \in \{0, 1\}, && \forall p \in P. && \text{(5.9f)}
\end{aligned}$$

The objective function minimizes, as in the previous models, the sum of sending costs and penalty costs. However for this formulation the penalty costs are activated whenever a negative path is selected. Constraints (5.9a) ensure only

one path is selected for each node, while constraints (5.9b) are on charge to eliminating any path $p$ if the starting node $s(p)$ is not a seed. Furthermore, constraints (5.9c) ensure that each node takes the shortest path among all activated nodes. Finally, constraints (5.9d) force each sub-path $p'$ of a path $p$ with $s(p') = s(p)$ to be in the solution whenever the path $p$ is in the solution. This last family of constraints is equivalent to constraints (5.3). Notice that this formulation has an exponential number of variables and constraints.

## 5.4 Formulation improvements

### 5.4.1 Valid inequalities

The formulations in the last subsections can be weak. One way to strengthen them is to introduce a set of valid inequalities to improve the associated linear relaxations. From the definition of the problem, the following constraints can be added to the formulations introduced in Section 5.3.

**Lemma 1.** *For each pair of nodes $k, h \in V$, such that $k \neq h$, the constraints:*

$$z_h^i \geq y_{ij}^k + z_h^k - 1, \qquad \forall (i,j) \in A, \tag{5.10}$$

$$z_h^i \leq z_h^k - y_{ij}^k + 1, \qquad \forall (i,j) \in A, \tag{5.11}$$

*are valid for the Max-Inf problem.*

*Proof.* From the definition of the problem, each non-seed node $k$ (i.e., $z_k^k = 0$), will receive the information through a shortest path from one seed node. To this end, if $h \in V$ is the seed of $k$ (i.e., $z_h^k = 1$), then, $h$ must be the seed of all nodes in the shortest path from $h$ to $k$. Therefore, on the one hand constraints (5.10) imposes to $z_h^i$ to be equal to 1 if $i$ is in the shortest path between the node $k$ and

her seed $h$ ($y_{ij}^k = 1$). On the other hand, constraints (5.11) avoid a node $i$ in the path to $k$ to be influenced by any seed node $h$ if $z_h^k = 0$. □

**Lemma 2.** *The inequality:*

$$\Pi_h^k \leq \Pi_i^k + M(2 - z_i^i - z_h^k), \qquad \forall i, k \in V, \tag{5.12}$$

*is valid for all feasible solutions of Bellman's formulation.*

*Proof.* If $i$ is a seed, and $h$ is the seed of $k$, then, $\Pi_h^k \leq \Pi_i^k$. This means that the length of the shortest path from $h$ to $k$ is not greater than the shortest paths from the other selected seeds. □

**Lemma 3.** *Let $\mathcal{P}^+$ be the set of all positive paths in the graph. For each $k \in V$, the set of inequalities:*

$$\sum_{k \in V} \sum_{(i,j) \in p} y_{ij}^k \leq |p| - (\pi_{d(p)} - \pi_{s(p)}), \qquad \forall p \in \mathcal{P}^+, \tag{5.13}$$

$$\sum_{k \in V} \sum_{(i,j) \in p} y_{ij}^k \leq |p| - (\pi_{s(p)} - \pi_{d(p)}), \qquad \forall p \in \mathcal{P}^+. \tag{5.14}$$

*are valid for the Max-Inf problem.*

*Proof.* Inequalities (5.13) and (5.14) are equivalent to:

$$\sum_{k \in V} \sum_{(i,j) \in p} y_{ij}^k \leq |p| - |\pi_{d(p)} - \pi_{s(p)}|, \qquad \forall p \in \mathcal{P}^+.$$

Two cases can happen:

$\pi_{s(p)} = \pi_{p(p)}$**:** In that case, $s(p)$ and $d(p)$ receive the same information and the inequality is redundant.

$\pi_{s(p)} \neq \pi_{p(p)}$**:** In that case, $s(p)$ and $d(p)$ receive opposite information, which implies the positive path $p$ cannot be in the solution.

□

**Lemma 4.** *Let $\mathcal{P}^-$ be the set of all negative paths in the graph. For each $k \in V$, the set of inequalities:*

$$\sum_{k \in V} \sum_{(i,j) \in p} y_{ij}^k \leq |p| + 1 - (\pi_{d(p)} + \pi_{s(p)}), \qquad \forall p \in \mathcal{P}^-, \tag{5.15}$$

$$\sum_{k \in V} \sum_{(i,j) \in p} y_{ij}^k \leq |p| - 1 + (\pi_{d(p)} + \pi_{d(p)}), \qquad \forall p \in \mathcal{P}^-. \tag{5.16}$$

*are valid for the max-inf problem.*

*Proof.* Inequalities (5.15) and (5.16) are equivalent to:

$$\sum_{k \in V} \sum_{(i,j) \in p} y_{ij}^k \leq |p| - |1 - \pi_{d(p)} - \pi_{s(p)}|, \qquad \forall p \in \mathcal{P}^-.$$

Two cases can happen:

$1 - \pi_{d(p)} - \pi_{s(p)} = 0$**:** In that case, $s(p)$ and $d(p)$ receive opposite information, then, the inequality is redundant.

$1 - \pi_{d(p)} - \pi_{s(p)} = 1$**:** In that case, $s(p)$ and $d(p)$ receive the same information, which implies the negative path $p$ cannot be in the solution.

□

### 5.4.2 Instance pre-processing

As we deal with social networks, real instances can be huge in terms of size [34; 85]. In the following, we propose some rules to reduce the size of an instance.

**Rule 1** (Arc-reduce)**.** *An arc $(i, j) \in A$ can be removed from the instance if at least one of the following conditions is satisfied:*

- *If there exists a path from $i$ to $j$ with length less than $d_{ij}$.*
- *If the arc $(i, j)$ does not belong to any shortest path between all pairs of nodes.*

We can also reduce the number of nodes in instance in some cases.

**Rule 2** (Node-reduce)**.** *A node $i \in V$ can be removed from the instance if one of the following conditions is satisfied:*

- *$i$ is an isolated node.*
- *The node $i$ does not have outgoing links (i.e., $N(i) = \{j \in V : (i, j) \in A\} = \emptyset$).*

If the shortest path between all pairs of nodes is known, we can fix some variables.

**Rule 3.** *The value of a variable $y_{ij}^k$ for $(i, j) \in A$ and $k \in V$ can be set as:*

- *$y_{ij}^k = 0$ if the arc $(i, j)$ does not belong to the tree defined by all shortest paths from the node $k$ to all nodes $k$.*
- *$y_{ij}^k = 1$ if the arc $(i, j)$ is included in the intersection of all shortest paths arriving to $k$.*

*Also, $z_k^k = 1$ if node $k \in V$ has only outgoing links (i.e., there is no node $i$ such that $(i, k) \in A$).*

## 5.5 Special case instances: balanced graphs

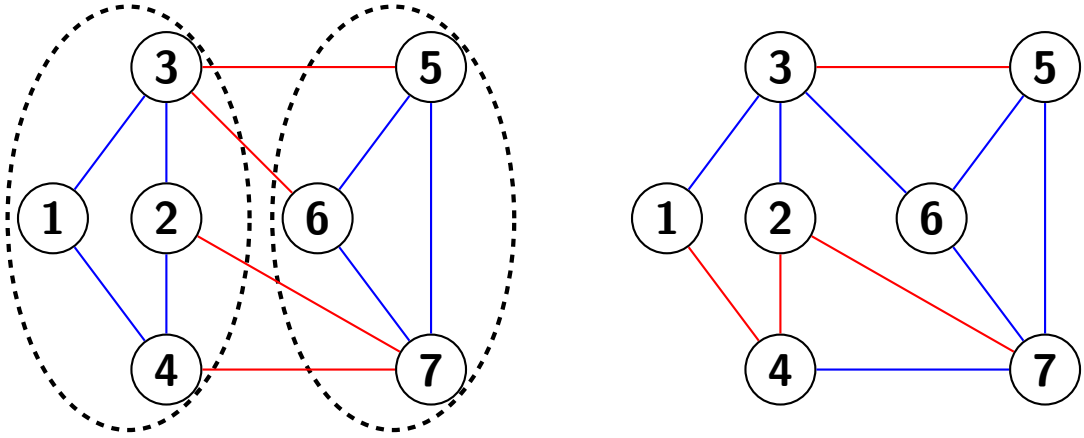
The proposed bi-level model for maximum influence problem is easy to solve in the case of unsigned networks (all the links are positive), where the optimal solution

amounts to select one seed with the least cost for each connected component. Then, each non-seed node will receive the information through the shortest path from the selected seed. Hence, the problem is a minimum spanning tree problem. However, in the general case the problem is $\mathcal{NP}$-hard and it is difficult to find an exact solution. In the literature many works studied balance in signed social networks [11; 59; 60; 96; 97; 98]. From these works, we can conclude that there exists many real instances that are balanced or almost. For this reason, in this section, we investigate the problem in special cases of networks for which we could solve the problem. Let us start by the definition of a balanced signed graph:

**Definition 3** (Balanced signed graphs:)**.** *Given an undirected signed graph $G(V, E)$ with the sets $E^+$ and $E^-$ of positive and negative edges, respectively. $G$ is called balanced if $V$ can be partitioned into two or more independent clusters $\{\mathcal{C}_1, \mathcal{C}_2, .., \mathcal{C}_L\}$ which contain only positive edges inside the clusters (i.e., $\{(i,j) : i,j \in \mathcal{C}_l\} \subset E^+, \forall l \in \{1, .., L\}$) and negative links connect the clusters (i.e., $\{(i,j) : i \in \mathcal{C}_{l_1}, j \in \mathcal{C}_{l_2}\} \subset E^-, \forall l_1 \neq l_2 \in \{1, .., L\}$). Let us refer to $\{\mathcal{C}_1, \mathcal{C}_2, .., \mathcal{C}_L\}$ as a balanced partition of $G$.*

Figure 5.2 illustrates the definition for a balanced signed graph. We show next some results concerning the optimal solution of the Max-Inf problem in particular cases of the instances. In the rest of the section, we assume that the distance $d_{ij}$ of all positive links are supposed to be short than the distance of any negative edge. Notice that finding the balanced partition of a signed networks is easy in the case of balanced graphs. Let us start by the case of complete balanced instances:

**Theorem 13** (Complete balanced signed graph)**.** *Let us suppose that the studied network is defined by a complete undirected signed graph $G$. If $G$ is a balanced graph,then the optimal solution of the maximum influence problem can be defined*




(a) Balanced signed graph (b) Unbalanced signed graph

Figure 5.2: Example of a balanced (a) and an unbalanced (b) signed graphs. Negative edges are red while positive edges are blue.

*by:*

$$Z^* = \sum_{l \in \{1,..,L\}} \min \left\{ \min_{i \in \mathcal{C}^l} f_i \quad , \quad \sum_{i \in \mathcal{C}^l} C_i \right\}$$

*where,* $\{\mathcal{C}_1, \mathcal{C}_2, .., \mathcal{C}_L\}$ *is a balanced partition of* $G$*.*

*Proof.* Given a complete balanced graph $G(V, E)$, let $\{\mathcal{C}_1, .., \mathcal{C}_L\}$ be a balanced partition of $V$ on $L$ clusters.

We can easily check that if $\min_{i \in \mathcal{C}^l} f_i \leq \sum_{i \in \mathcal{C}^l} C_i$, the solution:

$$\textbf{For all } l \in \{1, .., L\} : \begin{cases} z_h^h = 1 & \text{with: } h = \arg\min_{j \in \mathcal{C}_l} f_j, \\ \pi_i = 0 & \forall i \in \mathcal{C}_l, \\ z_h^i = 1 & \forall i \in \mathcal{C}_l \quad \text{with: } h = \arg\min_{j \in \mathcal{C}_l} f_j, \end{cases}$$

is a feasible solution for the problem, and the solution:

$$\textbf{For all } l \in \{1,..,L\} : \begin{cases} z_h^h = 0 & \forall h \in \mathcal{C}_l \\ \pi_i = 1 & \forall i \in \mathcal{C}_l \\ z_h^i = 1 & \forall i \in \mathcal{C}_l, h \in V \backslash \{\mathcal{C}_l\} \end{cases}$$

is also feasible if $\min_{i \in \mathcal{C}^l} f_i \geq \sum_{i \in \mathcal{C}^l} C_i$. Let $Z^*$ is the optimal value of the Max-Inf problem. Thus:

$$Z^* \leq \sum_{l \in L} \min \left\{ \min_{i \in \mathcal{C}^l} f_i \quad , \quad \sum_{i \in \mathcal{C}^l} C_i \right\}.$$

Let us suppose now that:

$$Z^* < \sum_{l \in L} \min \left\{ \min_{i \in \mathcal{C}^l} f_i \quad , \quad \sum_{i \in \mathcal{C}^l} C_i \right\},$$

then, there exists at least one cluster $\mathcal{C}^{l'}$ with a cost less than: $\min\{\min_{i \in \mathcal{C}^{l'}} f_i, \sum_{i \in \mathcal{C}^{l'}} C_i\}$. Hence, $z_i^i = 0, \ \forall i \in \mathcal{C}^{l'}$ (i.e., the cluster does not contain any seeds), and there exists at least one node positively influenced, which is impossible because of the balance of the graph, then:

$$Z^* \geq \sum_{l \in L} \min \left\{ \min_{i \in \mathcal{C}^l} f_i \quad , \quad \sum_{i \in \mathcal{C}^l} C_i \right\}.$$

As a result, we find:

$$Z^* = \sum_{l \in L} \min \left\{ \min_{i \in \mathcal{C}^l} f_i \quad , \quad \sum_{i \in \mathcal{C}^l} C_i \right\}.$$

□

This result can be generalized for a case of non complete balanced graphs.

**Theorem 14** (Balanced graphs with complete clusters)**.** *Let $G$ be a balanced signed graph. Let $\{\mathcal{C}_1, .., \mathcal{C}_L\}$ be a balanced partition such that $(C_l, E[C_l])$ is a complete graph, for each $l \in \{1, .., L\}$. The optimal value of the maximum influence problem defined on $G$ is:*

$$Z^* = \sum_{l \in L} \min \left\{ \min_{i \in \mathcal{C}^l} f_i \quad , \quad \sum_{i \in \mathcal{C}^l} C_i \right\}$$

*Proof.* The proof is the same done in Theorem 13. □

The newt result describe the optimal solution for another case of non-complete balanced graphs.

**Lemma 5.** *In the case of balanced graph with $L$ clusters and each node of a cluster is connected to all nodes of the other clusters, it is sufficient to find the optimal solution in each cluster without taking in consideration the other clusters.*

*Proof.* Since the graph is complete between the clusters, then each node of a cluster $l$ is a neighbour of a seed on another cluster. □

Thus, solving the problem in this case of instances can be reduced to find the solution on each cluster separately.

## 5.6 Numerical experiments

In subsections 5.3.2, 5.3.3, and 5.3.4 we presented five different formulations for Max-Inf problem defined on signed social networks. Four formulations are compact and can be solved by using MILP solvers.

In this section, we present two different experiments. First, we compare the proposed formulations. Then, numerical evaluations are performed to compare the improvement on linear relaxation and CPU time after introducing the valid

inequalities. All formulations are implemented in Julia 1.1.0, and the problems are solved using Gurobi 7.5.1 with default parameters and a time limit of 3600s. Simulations were performed on an Intel(R)core TM i7-6500M CPU@2.50 2.60GHz computer with 8 GB of RAM.

The different methods are tested on two sets of random instances. The first one concerns instances generated with no community structure. This set is generated by varying the number of nodes $n \in \{10, 30, 50, 100\}$, the density of graph $dens \in \{0.2, 0.5, 0.8\}$ and the percentage of negative edges in the graph $per \in \{20\%, 50\%, 80\%\}$. For each combination of $(n, dens, per)$, four instances were generated with different scenarios of costs and penalties. First, we fix the values $f_i = 5$ and $C_i = 25$ (resp, $f_i = 25$ and $C_i = 5$), and the distances are generated randomly in $[5, 25]$ . Then, we consider $f_i = C_i = 5$ and $d_{ij}$ are randomly generated in $[5, 25]$. Finally, all of $f_i$, $C_i$, and $d_{ij}$ are randomly generated in $[5, 25]$.

The second set concerns instances generated with some community structure. This set is generated with the same characteristics of the precedent instances, the only difference is that the instances here are balanced and contains $n/10$ clusters. Then, for each instance we shift the sign of the arc with a percentage in $\{0.2, 0.5, 0.8\}$.

We summarize next all methods used to solve the Max-Inf problem:

**KKT1:** One level formulation based on KKT optimality conditions and $\pi \in \Lambda_1$.

**KKT2:** One level formulation based on KKT optimality conditions and $\pi \in \Lambda_2$.

**Bellman1:** One level formulation based on Bellman's optimality conditions and $\pi \in \Lambda_1$.

**Bellman2:** One level formulation based on Bellman's optimality conditions and $\pi \in \Lambda_2$.

| *n-dens-per* | KKT1 | KKT2 | Bellman1 | Bellman2 | SP1 | SP2 |
|---|---|---|---|---|---|---|
| 10-0.2-0.2 | 0.08 | 0.09 | 0.08 | 0.15 | 0.21 | 0.22 |
| 10-0.5-0.2 | 0.09 | 0.11 | 0.03 | 0.05 | 0.08 | 0.09 |
| 10-0.8-0.2 | 0.08 | 0.08 | 0.03 | 0.03 | 0.55 | 0.41 |
| 10-0.2-0.5 | 0.08 | 0.14 | 0.03 | 0.05 | 0.06 | 0.05 |
| 10-0.5-0.5 | 0.14 | 0.23 | 0.05 | 0.06 | 0.58 | 0.58 |
| 10-0.8-0.5 | 0.05 | 0.06 | 0.05 | 0.05 | 0.69 | 0.58 |
| 10-0.2-0.8 | 0.06 | 0.09 | 0.03 | 0.06 | 0.06 | 0.11 |
| 10-0.5-0.8 | 0.05 | 0.06 | 0.03 | 0.05 | 0.44 | 0.75 |
| 30-0.2-0.2 | 3,600.04 | 3,600.04 | 1.95 | 5.34 | 45.27 | 186.15 |
| 30-0.5-0.2 | 3,600.12 | 3,600.11 | 5.59 | 10.67 | 26.23 | 186.15 |
| 30-0.8-0.2 | 3,600.12 | 3,600.12 | 1.33 | 3.86 | 122.19 | 173.08 |
| 30-0.2-0.5 | 3,600.07 | 3,600.05 | 9.31 | 19.39 | 40.44 | 66.58 |
| 30-0.5-0.5 | 3,600.08 | 3,600.05 | 6.50 | 9.72 | 362.41 | 166.23 |
| 30-0.8-0.5 | 150.82 | 154.65 | 8.31 | 6.00 | 114.23 | 185.37 |
| 30-0.2-0.8 | 3,600.07 | 3,600.08 | 21.49 | 122.88 | 79.24 | 71.91 |
| 30-0.5-0.8 | 232.23 | 427.75 | 18.47 | 13.33 | 108.81 | 121.68 |
| 30-0.8-0.8 | 149.97 | 159.33 | 12.38 | 6.15 | 96.95 | 153.07 |
| 50-0.2-0.2 | 3,600.56 | 3,600.15 | 60.59 | 217.08 | 3,600.10 | 3,600.12 |
| 50-0.5-0.2 | 2,668.48 | 3,600.12 | 47.73 | 42.18 | 7.45 | 7.50 |
| 50-0.8-0.2 | 3,600.44 | 3,600.18 | 50.26 | 20.82 | 12.61 | 12.66 |
| Average | 1600.18 | 1657.17 | 12.21 | 23.89 | 230.93 | 246.66 |

Table 5.1: Comparison of CPU time in second on the first set of random instances.

**SP1:** One level formulation based on shortest path constraints (5.7d) and $\pi \in \Lambda_1$.

**SP2:** One level formulation based on shortest path constraints (5.7d) and $\pi \in \Lambda_2$.

First, we compare the different methods with respect to solution times. The obtained results are shown in Tables 5.1 and 5.2 for the first and the second sets of random instances. In each row, we report the average CPU time in seconds spent by the different methods to get the optimal solutions of each one of the four instances with the same $(n, dens, per)$ values.

The results of tables 5.1 and 5.2 show that the CPU of the formulations **KKT**, **Bellman** , and **SP** are similar in both of methods to define the variable $\pi$. Also, we can see in the last column of the table, the time used to solve the

| n-d-% | KKT1 | KKT2 | Bellman1 | Bellman2 | SP1 | SP2 |
|---|---|---|---|---|---|---|
| 30-0.2-0.2 | 5.71 | 5.16 | 0.75 | 1.91 | 13.76 | 7.41 |
| 30-0.2-0.5 | 8.82 | 7.90 | 1.78 | 1.84 | 29.56 | 5.55 |
| 30-0.2-0.8 | 9.48 | 9.24 | 0.73 | 4.36 | 35.02 | 6.77 |
| 30-0.5-0.2 | 10.47 | 9.18 | 17.30 | 7.15 | 29.20 | 10.54 |
| 30-0.5-0.5 | 47.14 | 394.71 | 1.77 | 2.56 | 8.56 | 34.42 |
| 30-0.5-0.8 | 175.30 | 2,462.99 | 1.06 | 1.59 | 14.53 | 15.85 |
| 30-0.8-0.2 | 2.52 | 1,847.38 | 0.56 | 1.56 | 6.18 | 13.05 |
| 30-0.8-0.5 | 868.64 | 3,600.04 | 5.61 | 1.34 | 17.40 | 32.39 |
| 30-0.8-0.8 | 1,203.20 | 3,600.04 | 9.75 | 24.96 | 87.11 | 116.58 |
| 50-0.2-0.2 | 192.22 | 3,311.93 | 45.58 | 51.96 | 357.17 | 90.70 |
| 50-0.2-0.5 | 3,600.70 | 3,600.16 | 55.64 | 65.72 | 561.81 | 168.30 |
| 50-0.2-0.8 | 3,603.40 | 3,600.12 | 80.38 | 93.19 | 3,600.19 | 118.24 |
| Average | 823.02 | 1,870.74 | 18.41 | 21.51 | 396.71 | 51.65 |

Table 5.2: Comparison of CPU time in second on second set of random instances.

shortest path problem between all pairs of nodes in the graph. We can remark that this time is negligible compared with the instance solving time. Comparing now the cpu time of all methods, the algorithms based on Bellman's optimality conditions are faster than the other algorithms for the majority of instances. If we consider the total average we can see that, **Bellman1** and **Bellman2** are almost 2 times faster that the shortest path based methods **SP1** and **SP2** (10 times faster than **KKT1** and **KKT2**), respectively. This results can be justified by the number of integer variables in each formulation and also the linear relaxation gap. In addition, the valid inequalities presented in Section 5.4 are introduced to strengthen the precedent formulations. In the obtained results, the necessary time to find the optimal solution did not changed despite of the improvement of the linear relaxation after adding the valid inequalities. This is because the number of additional generated constraints.

# Chapter 6

# Conclusion and perspectives

In this chapter, we present a summary of the contributions proposed in this thesis. We also discuss some perspectives and future work.

## Fixed charge network design problem with shortest path constraints

In Chapter 3, we proposed two new BILP formulations for the fixed charge network design problem with shortest path constraints. The two models are combined with the iterative cutting plane (**CP1**, **CP3**) and a branch-and-cut (**B&C1**, **B&C2**) algorithms. Then, theoretical comparison is done between the proposed formulations. We further provided a valid inequality for the formulation of FCNDP-SPC whenever there are commodities sharing the origin and the destination.

All algorithms are tested on two types of data. We generated a set of 405 random instances classified in three groups according to their difficulty. The results show that the FCNDP-SPC is much easier when $0° \leq \alpha \leq 10°$ and it becomes very difficult for $40° \leq \alpha \leq 50°$. In these instances, the iterative cutting plane algorithms are almost equivalent and more efficient than the compact formulations and the branch-and-cut algorithms. This somewhat surprising result can be partly explained by the high number of inequalities generated by the

branch-and-cut algorithms and the time consumed by the separation problem.

We also used instances from the literature. Two real instances Ravenna (Italy) [23] and Albany, NY, (USA) [131] used to test the different algorithms. The obtained results show the time efficiency of our cutting plane algorithm (**CP1**) in comparison with the other algorithms.

From the experimental results, we can make two major observations. First, branch-and-cut and cutting plane approaches are better than the compact formulations for all types of instances. We saw that the iterative cutting plane algorithms outperform the branch-and-cut algorithm in terms of CPU time. This is because the latter consumes more time in the cuts generation and the separation problems in each integer node of the branching tree. Also, our **CP1** cutting plane method has the best results for all real instances. Second, the valid inequality generated for the Ravenna data improves the running time for all algorithms.

## Fixed charge network design problem with user-optimal flow

In Chapter 4, we have studied the problem of Energy-aware Traffic Engineering using a multi-path routing protocol (ETE-MPR) to minimize link capacity utilization in ISP backbone networks. To formulate the problem, we proposed a new bi-level optimization model where the objective of the upper-level problem is to activate network devices minimizing the energy consumption and the lower level represents a multi-path routing protocol. Then, we presented two one-level MILP formulations, respectively based on complementarity slackness optimality and on residual network conditions and a BILP formulation using unfeasible flow elimination constraint.

The obtained results show that the iterative cutting plane and branch-and-cut algorithms are close in terms of CPU time and outperform the compact

formulations. Furthermore, the residual network formulation obtained the optimal solution in CPU time faster than the formulation based on KKT conditions. Finally, our multi-path model has been compared with the existing single path approach on the same set of instances, and we found that using the multi-path leads to lower network energy consumption for more than 45% of instances and a reduction on the total traffic for 40% of instances.

## Maximum influence in signed social networks

Finally, Chapter 5 studied a version of the maximum influence problem defined on signed social networks named here Max-Inf problem. The objective of this problem is to select a set of users of a signed social network to diffuse an information and influence the rest of users so as to minimize sending cost and the penalty costs due to negative influence. We proposed a bi-level optimization model for this problem. Differently from the approaches in the literature, the propagation model is considered as a protocol where each user adopts the first arriving information.

Similarly to the previous chapters, we reformulated the bi-level programming model as a one level MILP models using different optimality conditions of the second level problem: KKT optimality conditions, Bellman's optimality conditions, and shortest path constraints. These new formulations have been strengthened by adding a set of valid inequalities. In addition, preprocessing techniques were discussed. Also, polynomial time solution was presented for particular cases of networks.

Computational experiments are performed using random instances to compare the different proposed formulations. The obtained results showed the efficiency of the formulation based on Bellman's optimality condition over the other formulations for the majority of instances.

We know that for huge size real instances, the MILP models are not able to find the optimal solution. To this end, for future works, we look to use preprocessing techniques and to combine the optimization models with the properties of signed networks to solve large instances. Also, the proposed methods can be improved using the constraints of path based formulation (5.9d) to cut infeasible solutions in the branching tree (branch-and-cut algorithm).

Furthermore, a generalization to other variants of the problem will be studied. For instance, if we suppose that the information retained by each user $k$ is a combination of information arriving by the first $m$ shortest paths to $k$.

# Bibliography

[1] H. Abrahamsson, B. Ahlgren, J. Alonso, A. Andersson, and P. Kreuger. A multi-path routing algorithm for ip networks based on flow optimisation. In *Proc. of Proceedings of the 3rd International Conference on Quality of Future Internet Services and Internet Charging and QoS Technologies*, pages 135–144. Springer-Verlag, 2002.

[2] B. Addis, A. Capone, G. Carello, L. G. Gianoli, and B. Sansò. Energy management through optimized routing and device powering for greener communication networks. *IEEE/ACM Trans. Netw.*, 22(1):313–325, 2014. 62, 63, 64, 65, 66

[3] N. Agin. Optimum seeking with branch and bound. *Management Science*, 13(4):B–176, 1966. 10

[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. 18, 25, 26, 27, 30, 35, 73

[5] E. Amaldi, M. Bruglieri, and B. Fortz. On the hazmat transport network design problem. In *Network Optimization*, pages 327–338. Springer, 2011. 15, 30

[6] E. Amaldi, A. Capone, S. Coniglio, and L. G. Gianoli. Energy-aware traffic

engineering with elastic demands and MMF bandwidth allocation. In *Proc. of IEEE CAMAD'13*, pages 169–174, 2013. 62, 63, 65, 66, 83

[7] F. Amato, V. Moscato, A. Picariello, and G. Sperlí. Diffusion algorithms in multimedia social networks: A novel model. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 85–103. Springer, 2018. 91

[8] A. S. G. Andrae and T. Edler. On global electricity usage of communication technology: Trends to 2030. *Challenges*, 6(1):117–157, 2015. 3, 62

[9] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. *Finding cuts in the TSP (A preliminary report)*, volume 95. Citeseer, 1995. 11

[10] D. Applegate, R. Bixby, W. Cook, and V. Chvátal. On the solution of traveling salesman problems. 1998. 11

[11] N. Arinik, R. Figueiredo, and V. Labatut. Multiple partitioning of multiplex signed networks: Application to european parliament votes. *Social Networks*, 2019. 110

[12] E. Balas. Letter to the editor—a note on the branch-and-bound principle. *Operations Research*, 16(2):442–445, 1968. 10

[13] R. Banner and A. Orda. Multipath routing algorithms for congestion minimization. *IEEE/ACM Trans. Netw.*, 15(2):413–424, April 2007. 63, 64

[14] J. F. Bard and J. T. Moore. An algorithm for the discrete bilevel programming problem. *Naval Research Logistics (NRL)*, 39(3):419–435, 1992. 15

[15] R. Batta and C. Kwon. *Handbook of OR/MS models in hazardous materials transportation*. Springer, 2013. 30

[16] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear programming and network flows*. John Wiley & Sons, 2011. 30, 34, 69

[17] R. Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958. 22

[18] O. Ben-Ayed, C. E. Blair, D. E. Boyce, and L. J. LeBlanc. Construction of a real-world bilevel linear programming model of the highway network design problem. *Annals of Operations Research*, 34(1):219–254, 1992. 12

[19] P. Bertier and B. Roy. Procédure de résolution pour une classe de problèmes pouvant avoir un caractère combinatoire. *Cahiers du Centre d'études de recherche opérationnelle*, 6:202–208, 1964. 10

[20] W. F. Bialas and M. H. Karwan. Two-level linear programming. *Management science*, 30(8):1004–1020, 1984. 12

[21] L. Bianco, M. Caramia, and S. Giordani. A bilevel flow model for hazmat transportation network design. *Transportation Research Part C: Emerging Technologies*, 17(2):175–196, 2009. 12, 15, 31

[22] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti. Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures. *IEEE Commun. Surveys Tuts.*, 13(2):223–244, 2011. 15, 62

[23] S. Bonvicini and G. Spadoni. A hazmat multi-commodity routing model satisfying risk criteria: A case study. *Journal of Loss Prevention in the Process Industries*, 21(4):345–358, 2008. 47, 58, 118

[24] I. Bouras, R. Figueiredo, M. Poss, and F. Zhoo. Exact Algorithms for Fixed Charge Network Design Problem with User-Optimal Flows. In *IWOBIP:*

*International Workshop on Bilevel Programming*, Lille, France, June 2018. xiv, 5

[25] I. Bouras, R. Figueiredo, M. Poss, and F. Zhou. Minimizing energy and link utilization in isp backbone networks with multi-path routing: a bi-level approach. *Optimization Letters*, pages 1–19. xiv, 6

[26] I. Bouras, R. Figueiredo, M. Poss, and F. Zhou. Exact Algorithms for Fixed Charge Network Design Problem with User Optimal Flow. In *Roadef 2018*, Lorient, France, Feb. 2018. xiv, 5

[27] I. Bouras, R. Figueiredo, M. Poss, and F. Zhou. Bi-level formulation for Minimizing Energy and Link Utilization in ISP Backbone Networks with Multipath Routing Protocol. In *ROADEF: Recherche Opérationnelle et d'Aide à la Décision*, Le Havre, France, Feb. 2019. LITIS and LMAH. xiv, 6

[28] I. Bouras, R. Figueiredo, M. Poss, and F. Zhou. On two new formulations for the fixed charge network design problem with shortest path constraints. *Computers & Operations Research*, 108:226–237, 2019. xiv, 5, 15, 77, 82, 102

[29] L. Brotcorne, S. Hanafi, and R. Mansi. A dynamic programming algorithm for the bilevel knapsack problem. *Operations Research Letters*, 37(3):215–218, 2009. 13

[30] L. Brotcorne, P. Marcotte, and G. Savard. Bilevel programming: The montreal school. *INFOR: Information Systems and Operational Research*, 46(4):231–246, 2008. 13

[31] J. J. Brown and P. H. Reingen. Social ties and word-of-mouth referral behavior. *Journal of Consumer research*, 14(3):350–362, 1987. 90

[32] W. Candler and R. Townsley. A linear two-level programming problem. *Computers & Operations Research*, 9(1):59–76, 1982. 12

[33] W. Chen, W. Lu, and N. Zhang. Time-critical influence maximization in social networks with time-delayed diffusion process. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012. 90

[34] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1029–1038. ACM, 2010. 90, 93, 108

[35] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009. 90

[36] X. Chen, A. Jukan, A. C. Drummond, and N. L. S. da Fonseca. A multi-path routing mechanism in optical networks with extremely high bandwidth requests. In *Proc. of IEEE Globecom'09*, pages 1–6, 2009. 63, 64, 65

[37] K.-Y. Chiang, N. Natarajan, A. Tewari, and I. S. Dhillon. Exploiting longer cycles for link prediction in signed networks. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1157–1162. ACM, 2011.

[38] L. Chiaraviglio, M. Mellia, and F. Neri. Minimizing isp network energy cost: Formulation and solutions. *IEEE/ACM Trans. Netw.*, 20(2):463–476, Apr. 2012. 62, 63

[39] M. Chiesa, G. Kindler, and M. Schapira. Traffic engineering with equal-cost-multipath: An algorithmic perspective. *IEEE/ACM Trans. Netw.*, 25(2):779–792, Apr. 2017. 63, 64, 65

[40] S.-W. Chiou. Bilevel programming for the continuous transport network design problem. *Transportation Research Part B: Methodological*, 39(4):361–383, 2005. 14

[41] A. Cianfrani, V. Eramo, M. Listanti, M. Polverini, and A. V. Vasilakos. An OSPF-Integrated Routing Strategy for QoS-Aware Energy Saving in IP Backbone Networks. *IEEE Trans. Netw. Service Manag.*, 9(3):254–267, Sept. 2012. 62, 63

[42] CISCO. CISCO Visual Networking Index. Technical report, February. 2019. 62

[43] S. Climer and W. Zhang. Cut-and-solve: An iterative search strategy for combinatorial optimization problems. *Artificial Intelligence*, 170(8):714 – 738, 2006.

[44] B. Colson, P. Marcotte, and G. Savard. Bilevel programming: A survey. *4or*, 3(2):87–107, 2005. 13

[45] W. Cook and J. L. Rich. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical report, 1999. 11

[46] F. Dabaghi, Z. Movahedi, and R. Langar. A survey on green routing protocols using sleep-scheduling in wired networks. *J. Netw. Comput. Appl.*, 77(C):106–122, Jan. 2017. 62

[47] S. Dempe. *Bilevel programming: A survey.* Dekan der Fak. für Mathematik und Informatik, 2003. 15

[48] S. Dempe, V. Kalashnikov, G. Perez-Valdes, and N. Kalashnykova. *Bilevel Programming Problems: Theory, Algorithms and Applications to Energy Networks*. Springer, 2015. 12, 34

[49] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354, 1992.

[50] B. Detienne. A mixed integer linear programming approach to minimize the number of late jobs with and without machine availability constraints. *European Journal of Operational Research*, 235(3):540–552, 2014.

[51] B. Detienne, L. Peridy, E. Pinson, and D. Rivreau. Génération de coupes pour la planification d'agents. Int. Conference Modélisation et Simulation MOSIM, 2006.

[52] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959. 19, 20

[53] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002. 56

[54] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66. ACM, 2001. 90

[55] E. Erkut and O. Alp. Designing a road network for hazardous materials shipments. *Computers & Operations Research*, 34(5):1389–1405, 2007. x, 2, 31

[56] E. Erkut and F. Gzara. Solving the hazmat transport network design problem. *Computers & Operations Research*, 35(7):2234–2247, 2008. 31, 69

[57] E. Even-Dar and A. Shapira. A note on maximizing the spread of influence in social networks. In *International Workshop on Web and Internet Economics*, pages 281–286. Springer, 2007. 90

[58] J.-B. Eytard. *A tropical geometry and discrete convexity approach to bilevel programming: application to smart data pricing in mobile telecommunication networks*. PhD thesis, Paris Saclay, 2018. 14

[59] R. Figueiredo, Y. Frota, and M. Labbé. Solution of the maximum k-balanced subgraph problem. In *International Conference on Learning and Intelligent Optimization*, pages 266–271. Springer, 2013. 110

[60] R. Figueiredo, Y. Frota, and M. Labbé. A branch-and-cut algorithm for the maximum k-balanced subgraph of a signed graph. *Discrete Applied Mathematics*, 261:164–185, 2019. 110

[61] R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962. 23

[62] L. R. Ford Jr. Network flow theory. Technical report, Rand Corp Santa Monica Ca, 1956. 22

[63] J. P. Forgas and S. M. Laham. Halo effects. *Cognitive illusions: Intriguing phenomena in judgement, thinking and memory*, pages 276–290, 2016. 93

[64] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987. 20

[65] X. Gandibleux, G. Soleilhac, A. Przybylski, F. Lucas, S. Ruzika, and P. Halffmann. voptsolver, a "get and run" solver of multiobjective linear

optimization problems built on julia and jump. *MCDM2017: 24th International Conference on Multiple Criteria Decision Making*, July 10-14, 2017. 88

[66] K. Genova and V. Guliashki. Linear integer programming methods and approaches–a survey. *Journal of Cybernetics and Information Technologies*, 11(1), 2011. 9

[67] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters*, 12(3):211–223, 2001. 90

[68] J. Goldenberg, B. Libai, and E. Muller. Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata. *Academy of Marketing Science Review*, 9(3):1–18, 2001. 90

[69] R. E. Gomory. An algorithm for integer solutions to linear programs. *Recent advances in mathematical programming*, 64(260-302):14, 1963. 11

[70] R. E. Gomory et al. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical society*, 64(5):275–278, 1958. 11

[71] M. Gondran and M. Minoux. *Graphs and algorithms*. Wiley, 1984. 20

[72] P. H. González, L. Simonetti, P. Michelon, C. Martinhon, and E. Santos. A variable fixing heuristic with local branching for the fixed charge uncapacitated network design problem with user-optimal flow. *Computers & Operations Research*, 76:134–146, 2016. 30, 49

[73] P. H. González, L. G. Simonetti, C. A. de Jesus Martinhon, E. Santos, and P. Y. P. Michelon. An improved relax-and-fix algorithm for the fixed charge network design problem with user-optimal flow. In *Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems*, pages 100–107, 2014. 15

[74] A. Goyal, W. Lu, and L. V. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *2011 IEEE 11th international conference on data mining*, pages 211–220. IEEE, 2011. 90

[75] M. Granovetter. Threshold models of collective behavior. *American journal of sociology*, 83(6):1420–1443, 1978. 90

[76] G. D. Greenwade. The Comprehensive Tex Archive Network (CTAN). *TUGBoat*, 14(3):342–351, 1993.

[77] M. Grötschel, A. Martin, and R. Weismantel. Packing steiner trees: A cutting plane algorithm and computational results. *Mathematical Programming*, 72(2):125–145, 1996. 11

[78] M. Gupta and S. Singh. Greening of the internet. In *Proc. of ACM SIGCOMM'03*, pages 19–26, 2003. 62

[79] F. Gzara. A cutting plane approach for bilevel hazardous material transport network design. *Operations Research Letters*, 41(1):40–46, 2013. xiii, 5, 31, 36, 38

[80] J. He and W. Song. Achieving near-optimal traffic engineering in hybrid software defined networks. In *Proc. of IFIP Networking'05*, pages 1–9, May 2015. 64

[81] M. M. Islam. *Development of Methods for Solving Bilevel Optimization Problems*. PhD thesis, The University of New South Wales Australia 11, 2018.

[82] D. S. Johnson and M. R. Garey. *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman, 1979. 9

[83] M. Jünger, G. Nemhauser, L. Wolsey, A. Schrijver, and R. Gomory. Cutting plane algorithms for integer programming. 1998. 11

[84] B. Y. Kara and V. Verter. Designing a road network for hazardous materials transportation. *Transportation Science*, 38(2):188–196, 2004. 15, 31, 32, 33, 34

[85] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003. 90, 93, 108

[86] D. Kempe, J. Kleinberg, and É. Tardos. Influential nodes in a diffusion model for social networks. In *International Colloquium on Automata, Languages, and Programming*, pages 1127–1138. Springer, 2005. 90

[87] J. Kim, W. Lee, and H. Yu. Ct-ic: Continuously activated and time-restricted independent cascade model for viral marketing. *Knowledge-Based Systems*, 62:57–68, 2014. 90

[88] M. Kimura and K. Saito. Tractable models for information diffusion in social networks. In *European conference on principles of data mining and knowledge discovery*, pages 259–271. Springer, 2006. 90

[89] J. Kleinberg and E. Tardos. *Algorithm design*. Pearson Education India, 2006. 19

[90] B. Korte and J. Vygen. *Combinatorial optimization*, volume 2. Springer. 20, 22, 23

[91] S. J. Lachman and A. R. Bass. A direct study of halo effect. *The journal of psychology*, 119(6):535–540, 1985. 93

[92] S. Lambert, W. V. Heddeghem, W. Vereecken, B. Lannoo, D. Colle, and M. Pickavet. Worldwide electricity consumption of communication networks. *Opt. Express*, 20(26):B513–B524, Dec. 2012. xi, 3, 62

[93] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966. 10

[94] L. J. LeBlanc and D. E. Boyce. A bilevel programming algorithm for exact solution of the network design problem with user-optimal flows. *Transportation Research Part B: Methodological*, 20(3):259 – 265, 1986. 30

[95] G. M. Lee and J. S. Choi. A survey of multipath routing for traffic engineering. *Information and Communications University*, pages 1–27, Jul. 2018. 63

[96] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 641–650. ACM, 2010. 110

[97] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1361–1370. ACM, 2010. 110

[98] M. Levorato, R. Figueiredo, Y. Frota, and L. Drummond. Evaluating balancing on social networks through the efficient solution of correlation clustering problems. *EURO Journal on Computational Optimization*, 5(4):467–498, 2017. 110

[99] M. Li, A. Lukyanenko, Z. Ou, A. Ylä-Jääski, S. Tarkoma, M. Coudron, and S. Secci. Multipath transmission for the internet: A survey. *IEEE Commun. Surveys Tuts.*, 18(4):2887–2925, 2016. 63

[100] Y. Li, W. Chen, Y. Wang, and Z.-L. Zhang. Influence diffusion dynamics and influence maximization in social networks with friend and foe relationships. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 657–666. ACM, 2013. 91

[101] Y. Li, W. Chen, Y. Wang, and Z.-L. Zhang. Voter model on signed social networks. *Internet Mathematics*, 11(2):93–133, 2015. 91

[102] X. Lin, Q. Jiao, and L. Wang. Opinion propagation over signed networks: Models and convergence analysis. *IEEE Transactions on Automatic Control*, 2018. 91

[103] B. Liu, G. Cong, D. Xu, and Y. Zeng. Time constrained influence maximization in social networks. In *2012 IEEE 12th international conference on data mining*, pages 439–448. IEEE, 2012. 90

[104] X. Liu, S. Mohanraj, M. Pioro, and D. Medhi. Multipath routing from a traffic engineering perspective: How beneficial is it? In *Proc. of IEEE ICNP'14*, pages 143–154, Oct 2014. 63, 64, 65

[105] A. Madkour, W. Aref, F. Rehman, A. Rahman, and S. Basalamah. A survey of shortest-path algorithms. 05 2017. 18

[106] T. L. Magnanti and R. T. Wong. Network design and transportation planning: Models and algorithms. *Transportation science*, 18(1):1–55, 1984. 30

[107] P. Marcotte. A note on a bilevel programming algorithm by leblanc and boyce. *Transportation Research Part B: Methodological*, 22(3):233 – 236, 1988. 30

[108] R. Mathieu, L. Pittard, and G. Anandalingam. Genetic algorithm based approach to bi-level linear programming. *RAIRO-Operations Research*, 28(1):1–21, 1994. 12

[109] A. Mauttone, M. Labbé, and R. Figueiredo. A tabu search approach to solve a network design problem with user-optimal flows. In *ALIO/EURO Workshop on Applied Combinatorial Optimization*, 2007. 69

[110] A. Mauttone, M. Labbé, and R. Figueiredo. A tabu search approach to solve a network design problem with user-optimal flows. In *VI ALIO/EURO Workshop on Applied Combinatorial Optimization*, 2008. 15, 30, 31, 35, 49

[111] P. Merindol, J. J. Pansiot, and S. Cateloin. Improving load balancing with multipath routing. In *Proc. of IEEE ICCCN'08*, pages 1–8, Aug. 2008. 64

[112] A. Migdalas, P. M. Pardalos, and P. Värbrand. *Multilevel optimization: algorithms and applications*, volume 20. Springer Science & Business Media, 2013. 13

[113] J. T. Moore and J. F. Bard. The mixed integer linear bilevel programming problem. *Operations research*, 38(5):911–921, 1990. 15

[114] D. Nace. A linear programming based approach for computing optimal fair

splittable routing. In *Proceedings ISCC 2002 Seventh International Symposium on Computers and Communications*, pages 468–474. IEEE, 2002.

[115] D. Nace and M. Pióro. Max-min fairness and its applications to routing and load-balancing in communication networks: A tutorial. *IEEE Commun. Surveys Tuts.*, 10(1-4):5–17, 2008.

[116] G. L. Nemhauser and G. Sigismondi. A strong cutting plane/branch-and-bound algorithm for node packing. *Journal of the Operational Research Society*, 43(5):443–457, 1992. 11

[117] I. Nishizaki and M. Sakawa. Stackelberg solutions to multiobjective two-level linear programming problems. *Journal of Optimization Theory and Applications*, 103(1):161–182, 1999. 12

[118] 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. http://sndlib.zib.de, extended version accepted in Networks, 2009. xxi, 83, 84

[119] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991. 11

[120] S.-L. Phouratsamay, S. Kedad-Sidhoum, and F. Pascual. Two-level lot-sizing with inventory bounds. *Discrete Optimization*, 30:1–19, 2018. 13

[121] P. Pisciella. On the reformulation of a particular class of bilevel problems. 2011. 12, 13, 16

[122] M. Poss. *Models and algorithms for network design problems.* PhD thesis, Université Libre de Bruxelles, 2011. 9

[123] V. J. Rayward-Smith, S. Rush, and G. P. McKeown. Efficiency considerations in the implementation of parallel branch-and-bound. *Annals of Operations Research*, 43(2):123–145, 1993. 10

[124] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70. ACM, 2002. 90

[125] M. Sakarovitch. *Optimisation combinatoire: méthodes mathématiques et algorithmiques: programmation discrète.* Hermann, 1984. 20

[126] C. Shen, R. Nishide, I. Piumarta, H. Takada, and W. Liang. Influence maximization in signed social networks. In *International Conference on Web Information Systems Engineering*, pages 399–414. Springer, 2015. 91

[127] S. K. Singh, T. Das, and A. Jukan. A survey on internet multipath routing and provisioning. *IEEE Commun. Surveys Tuts.*, 17(4):2157–2175, 2015. 63

[128] N. R. Suri and Y. Narahari. Determining the top-k nodes in social networks using the shapley value. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1509–1512. International Foundation for Autonomous Agents and Multiagent Systems, 2008. 90

[129] P. Toth and D. Vigo. *The vehicle routing problem.* SIAM, 2002. 11

[130] I. Toumazis and C. Kwon. Routing hazardous materials on time-dependent networks using conditional value-at-risk. *Transportation research part C: emerging technologies*, 37:73–92, 2013.

[131] I. Toumazis, C. Kwon, and R. Batta. Value-at-risk and conditional value-at-risk minimization for hazardous materials routing. In *Handbook of OR/MS Models in Hazardous Materials Transportation*, pages 127–154. Springer, 2013. 47, 60, 118

[132] W. van Ackooij, J. D. Boeck, B. Detienne, S. Pan, and M. Poss. Optimizing power generation in the presence of micro-grids. *European Journal of Operational Research*, 271(2):450–461, 2018. 16

[133] V. Verter and B. Y. Kara. A path-based approach for hazmat transport network design. *Management Science*, 54(1):29–40, 2008. 31

[134] L. Vicente, G. Savard, and J. Judice. Discrete linear bilevel programming problem. *Journal of optimization theory and applications*, 89(3):597–614, 1996. 14, 15

[135] H. von Stackelberg. *The Theory of the Market Economy*. Oxford University Press, 1952. 12

[136] J. Y. Wang, M. Ehrgott, K. N. Dirks, and A. Gupta. A bilevel multi-objective road pricing model for economic, environmental and health sustainability. *Transportation Research Procedia*, 3:393–402, 2014. 14

[137] L. Wang, F. Zhang, K. Zheng, A. V. Vasilakos, S. Ren, and Z. Liu. Energy-efficient flow scheduling and routing with hard deadlines in data center networks. In *2014 IEEE 34th International Conference on Distributed Computing Systems*, pages 248–257. IEEE, 2014. 78

[138] W. Wang and W. N. Street. Modeling and maximizing influence diffusion in social networks for viral marketing. *Applied network science*, 3(1):6, 2018. 90

[139] Y. Wang, G. Cong, G. Song, and K. Xie. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1039–1048. ACM, 2010. 91

[140] S. Warshall. A theorem on boolean matrices. In *Journal of the ACM*. Citeseer, 1962. 23

[141] L. A. Wolsey. *Integer programming*. Wiley-Interscience, New York, NY, USA, 1998. 8, 9, 21

[142] J. Zhang, K. Xi, and H. J. Chao. Load balancing in ip networks using generalized destination-based multipath routing. *IEEE/ACM Trans. Netw.*, 23(6):1959–1969, Dec. 2015. 63, 64

[143] X. Zhang, W. Wang, P. O. de Pablos, J. Tang, and X. Yan. Mapping development of social media research through different disciplines: Collaborative learning in management and computer science. *Computers in Human Behavior*, 51:1142–1153, 2015. 90