POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Design of robust networks.
Application to the design of wind farm cabling networks.

THOMAS RIDREMONT

Département de mathématiques et de génie industriel

Polytechnique Montréal

et

Conservatoire National des Arts et Métiers

Thèse en cotutelle présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques

Avril 2019

© Thomas Ridremont, 2019.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Design of robust networks.
Application to the design of wind farm cabling networks.

présentée par **Thomas RIDREMONT**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Sourour ELLOUMI, présidente
Alain HERTZ, membre et directeur de recherche
Marie-Christine COSTA, membre et codirectrice de recherche
Cédric BENTZ, membre et codirecteur de recherche
Miguel ANJOS, membre
Frédéric ROUPIN, membre
Dritan NACE, membre
Dominique QUADRI, membre externe

ACKNOWLEDGEMENTS

First of all, I would like to thank my thesis directors: Marie-Christine Costa, Cédric Bentz and Alain Hertz. Thank you for sharing with me your passion for operations research. Thank you for being present, patient, available, for accompanying me in my research project, with good humour, enthusiasm and kindness. Thank you to Marie-Christine and Cédric for being particularly present and responding at the end of a thesis where I particularly solicited them. Thank you to Alain for welcoming me perfectly to Montreal, for being very available, even when an ocean separated us.

Thanks to Dominique Quadri and Miguel Anjos, who did me the honour of being the rapporteurs of my thesis. Thank you for your time, interest and comments. Thank you also to Sourour Elloumi, Dritan Nace and Frédéric Roupin for the honour of serving on my thesis jury.

Thank you to the people of CEDRIC for their advice, their good humour, their help but above all the good working atmosphere, as well as to the members of GERAD and UMA. I am thinking of Amélie, Agnès, Alain B., Alain F., Christophe, Daniel, Éric, Maurice, Safia, Sami, Sourour, Stéphane and Zacharie, among others. Thank you to all those with whom I was able to share my office, I think in particular of Pierre-Louis and Dimitri, who welcomed me and with whom I learned a lot, thank you to Arnaud, whom I welcomed, and to Antoine and Hadrien.

Thank you to my friends in Montreal, who made me feel at home there. Thanks to my friends from Angers and Paris too, who have been present during these few years, thanks to Brice and Sylvain for welcoming me with open arms at several times. A special thank you to my friend Florian, who was very supportive throughout my thesis and with whom I was able to discuss the life of a doctoral student. Thank you to Camille, who supported me in the last moments of the thesis, thanks for her patience, tenacity, attention and especially her understanding.

Finally, a big thank you to my family, my grandparents and my grandmother. Thank you to them who have been able to support me throughout this period, to help me when I needed it, especially at the end of this thesis. Thanks to Anaëlle and Clara, for their good humour. Thank you to my brother Damien, with whom I shared a lot and who always listened to me. Thank you to my father, who was able to give me his advice, and thank you to my mother, who is always a great help and who gave me a lot of time.

RÉSUMÉ

Aujourd'hui, la conception de réseaux est une problématique cruciale qui se pose dans beaucoup de domaines tels que le transport ou l'énergie. En particulier, il est devenu nécessaire d'optimiser la façon dont sont conçus les réseaux permettant de produire de l'énergie. On se concentre ici sur la production électrique produite à travers des parcs éoliens. Cette énergie apparait plus que jamais comme une bonne alternative à la production d'électricité via des centrales thermiques ou nucléaires.

Nous nous intéressons dans cette thèse à la conception du câblage collectant l'énergie dans les parcs éoliens. On connaît alors la position de l'ensemble des éoliennes appartenant au parc ainsi que celle du site central collecteur vers laquelle l'énergie doit être acheminée. On connaît également la position des câbles que l'on peut construire, leurs capacités, et la position des nœuds d'interconnexion possibles. Il s'agit de déterminer un câblage de coût minimal permettant de relier l'ensemble des éoliennes à la sous-station, tel que celui-ci soit résistant à un certain nombre de pannes sur le réseau.

Mots clés: Recherche opérationnelle, Optimisation combinatoire, Conception de réseaux robustes, Théorie des graphes, Programmation en nombres entiers, Câblage de parcs éoliens.

ABSTRACT

Nowadays, the design of networks has become a decisive problematic which appears in many fields such as transport or energy. In particular, it has become necessary and important to optimize the way in which networks used to produce, collect or transport energy are designed. We focus in this thesis on electricity produced through wind farms. The production of energy by wind turbines appears more than ever like a good alternative to the electrical production of thermal or nuclear power plants, giving that both of those production can have harmful consequences on the environment. It has then become necessary to optimize the design and construction of such networks.

We focus in this thesis on the design of the cabling network which allows to collect and route the energy from the wind turbines to a sub-station, linking the wind farm to the electrical network. In this problem, we know the location of each wind turbine of the farm and the one of the sub-station. We also know the location of possible inter-connection nodes which allow to connect different cables between them. Each wind turbine produces a known quantity of energy and with each cable are associated a cost and a capacity (the maximum amount of energy that can be routed through this cable). The optimization problem that we consider is to select a set of cables of minimum cost such that the energy produced from the wind turbines can be routed to the sub-station in the network induced by this set of cables, without exceeding the capacity of each cable. We focus on cabling networks resilient to breakdowns.

Keywords : Operations Research, Combinatorial optimization, Robust networks design, Graph theory, Mixed integer programming, Wind farm cabling networks.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

RÉSUMÉ iv

ABSTRACT v

TABLE OF CONTENTS vi

LIST OF TABLES ix

LIST OF FIGURES x

CHAPTER 1 INTRODUCTION 1
- 1.1 General introduction 1
- 1.2 Preliminaries 4
  - 1.2.1 General notions in graph theory 4
  - 1.2.2 Flows and networks 5
  - 1.2.3 Steiner trees and networks 9
- 1.3 Previous work 10
  - 1.3.1 Wind farm cable layout optimization 10
  - 1.3.2 Steiner arborescence problems 11
  - 1.3.3 Robust Steiner networks 12
  - 1.3.4 The k most vital arcs in flow networks and network interdiction problems 14

CHAPTER 2 ROBUST ARBORESCENCES 16
- 2.1 Introduction 16
- 2.2 Definition of problems and complexity results 16
- 2.3 Mathematical formulations and tests 20

CHAPTER 3 CAPACITATED ROOTED k-EDGE CONNECTED STEINER NETWORK PROBLEM (**CRkECSN**) 31
- 3.1 Definitions and notations 31
- 3.2 Formulations 32
  - 3.2.1 Cutset formulation 32
  - 3.2.2 Flow formulation 36

3.2.3 Bilevel formulation 37
3.3 Relations between the formulations 44
3.3.1 Relations between the bilevel and the cutset formulations 44
3.3.2 Relations between the flow and the cutset formulations 48
3.4 Addition of protected arcs 49
3.4.1 Cut-set formulation 50
3.4.2 Equivalence between cut-set and bilevel formulations in the protected case 52
3.4.3 Flow formulation 53
3.5 Valid and strengthening inequalities 53
3.5.1 Case without the possibility of protecting arcs 54
3.5.2 Case with the possibility of protecting arcs 55
3.6 Results analysis 56

CHAPTER 4 A TABU SEARCH FOR THE DESIGN OF CAPACITATED ROOTED k-EDGE CONNECTED STEINER PLANAR NETWORKS 68
4.1 Introduction 68
4.2 Testing survivability 68
4.3 Tabu Search 73
4.3.1 A repair procedure 75
4.3.2 Inclusion-wise minimal solutions 77
4.3.3 The proposed tabu search for the CRkECSN 79
4.4 Computational Experiments 81

CHAPTER 5 WIND FARM CABLE LAYOUT OPTIMIZATION WITH CONSTRAINTS OF LOAD FLOW AND ROBUSTNESS 89
5.1 Introduction 89
5.1.1 Presentation of the problem 89
5.1.2 The Load Flow constraints 90
5.2 The problem without breakdowns 94
5.2.1 Mathematical formulation 94
5.2.2 Variables and constraints 95
5.2.3 Mathematical program 97
5.2.4 Linearization of the program 98
5.3 Robust approach 99
5.3.1 The case $k = 1$ 99
5.3.2 Bilevel formulation 103

5.4 Results analysis 105

CONCLUSION AND RECOMMENDATIONS 107

LIST OF TABLES

Table 2.1: Results on robust arborescences and data parameters 26
Table 3.1: Instance parameters 56
Table 3.2: Results for non-uniform capacities, $k = 1$ and $k' = 0$ 58
Table 3.3: Results for non-uniform capacities, $k = 2$ and $k' = 0$ 59
Table 3.4: Results for non-uniform capacities, $k = 3$ and $k' = 0$ 60
Table 3.5: Results for non-uniform capacities, $k = 1$ and $k' \in \{1, 2, 3\}$ 61
Table 3.6: Results for non-uniform capacities, $k = 2$ and $k' \in \{1, 2, 3\}$ 61
Table 3.7: Results for non-uniform capacities, $k = 3$ and $k' \in \{1, 2, 3\}$ 63
Table 3.8: Results for uniform capacities and $k = 1$ 64
Table 3.9: Results for uniform capacities and $k = 2$ 65
Table 3.10: Results for uniform capacities and $k = 3$ 66
Table 4.1: Results for $k = 1$ where $(i, j) \in A$ if and only if $(j, i) \in A$. 85
Table 4.2: Results for $k = 2$ where $(i, j) \in A$ if and only if $(j, i) \in A$. 86
Table 4.3: Results for $k = 3$ where $(i, j) \in A$ if and only if $(j, i) \in A$. 86
Table 4.4: Results for $k = 1$ where $(i, j) \notin A$ if $(j, i) \in A$. 87
Table 4.5: Results for $k = 2$ where $(i, j) \notin A$ if $(j, i) \in A$. 88
Table 4.6: Results for $k = 3$ where $(i, j) \notin A$ if $(j, i) \in A$. 88
Table 5.1: Results of the tests for the non-robust case and for the robust case with $k = 1$ 106

LIST OF FIGURES

Figure 1.1: Example of graph transformation for the vertex v 9

Figure 2.1: Graph and **RSpAF** solution resulting from the 3-Partition instance in which $m = 2$, $B = 11$, $D = \{5, 3, 4, 3, 4, 3\}$ 20

Figure 2.2: Resulting arborescence for the fourth data set for $CStA$ 27

Figure 2.3: Resulting arborescence for the fourth data set for $RCStA$ 28

Figure 2.4: Resulting arborescence for the fourth data set for $BRCStA$ 29

Figure 2.5: Resulting arborescence for the fourth data set for $BRCStA_{bounded-robust-cost}$ 30

Figure 3.1: Example of addition of a sink to the input graph 33

Figure 3.2: Graph where Inequalities (3.24a) and (3.24b) cut some non-integer solutions 54

Figure 3.3: Cost of the solutions for different instances 67

Figure 4.1: Construction of $\widetilde{G'}$ from G' 69

Figure 4.2: justification=centering 70

Figure 4.3: Construction of $2D_{\widetilde{G'}}$ for the graph $\widetilde{G'}$ of Figure 4.1 72

Figure 5.1: A network buildt on a grid 4×3 95

CHAPTER 1 INTRODUCTION

1.1 General introduction

In the 21st century, it appears that the biggest challenge that will face our population is the global warming. The air temperature has been increased by 1.5 degrees since preindustrial era and this rise has been linked to human activities. Almost all specialists among the community agree that the major cause of this global warming can be attributed to the increasing production of greenhouse gas (caused by carbon monoxide emission or methane principally). The consequences could be diverse and terrible: for the climate (for example extreme heat in some parts of the globe or increase of extreme weather events like storms, floods, cyclones and droughts); for the ecosystem and the biodiversity (increase of ocean levels, destruction of fragile ecosystems like coral reef and Amazon rainforest and several extinctions of species); on our society and its economy (infrastructures to adapt like medical ones or housings, public health and capacity to feed the population). Many forces appear to fight the global warming and involve ecology in our way of life (reducing our consumption of energy, limiting the food waste, optimizing the management of resources, avoiding to consume products with a high carbon print). Sustainable development, which aims to exploit natural and biological resources at a rhythm which does not lead impoverishment or even exhaustion but makes possible the sustain of biological productivity of resources in the biosphere, comes out as a valid orientation in order to limit those consequences. Regarding the electrical production, wind farms, photo-voltaic panels and hydro-electrical facilities appear to be interesting directions in order to reduce greenhouse gas emissions.

Nowadays, the design of networks has become a decisive problematic which appears in many fields such as transport, telecommunications or energy. In particular, it has become important and even necessary to optimize the way in which networks used to produce, collect or transport energy are designed. We are interested in this thesis on electricity produced through wind farms. The production of energy by wind turbines appears more than ever like a good alternative to the electrical production of thermal or nuclear power plants, giving that both of those productions can have harmful consequences on the environment. The development of wind farms is then a global issue and hence it has become necessary to optimize the design and construction of such networks.

We focus in this thesis on the design of the cabling network which allows to collect and route

the energy from the wind turbines to a sub-station, linking the wind farm to the electrical network. In this problem, we know the location of each wind turbine of the farm and the one of the sub-station. We also know the location of possible inter-connection nodes which allow to connect different cables between them. Each wind turbine produces a known quantity of energy and with each cable are associated a cost and a capacity (the maximum amount of energy that can be routed through this cable). The optimization problem that we consider is to select a set of cables of minimum cost such that the energy produced from the wind turbines can be routed to the sub-station in the network induced by this set of cables, without exceeding the capacity of each cable. Hence there must exist a path using cables between each turbine and the substation, but not necessarily with inter-connection nodes, which are optional points in the network.

In this context, breakdowns can occur on cables or devices of the network (caused by the environment or by a problem with a turbine or inter-connexion node for example). We focus more precisely on the design of robust (or resilient) networks for several robust notions that will be defined in this thesis. We take into account some data incertitude: we consider the case of breakdowns on cables or nodes once the network is built (in this thesis we focus on breakdowns on cables, but breakdowns on nodes can be reduced to breakdowns on cables after a transformation of the graph). We then aim to minimize the cost of the network to build while respecting robustness constraints allowing to limit the damaging repercussions in case of a breakdown on one or several cables in the network.

In the context of this thesis, we have been in contact with EDF (Électricité de France), first producer and supplier of electricity in France, via PGMO (Programme Gaspard Monge pour l'Optimisation de la Fondation Mathématique Jacques Hadamard) and engineers working in the field of renewable energy networks. It appears that combinatorial and discrete aspects of those problems have been sparsely studied until now at EDF. Although the design of the cabling networks presents high economic stakes, robustness and resilience to breakdowns are important criteria too. Some work has also been done with the Canadian company Hatch, which led us to test our work on real data (we can underline that the French wind farms we have been working on with EDF are offshore whereas Canadian wind farms are onshore). In offshore environment, each wind turbine produces about the same quantity of electricity, so we can make the assumption that the energy produced by the wind turbines is uniform.

In this thesis, we alternate between more theoretical problems related to the design of wind

farm cabling networks by reformulating the set of electricity constraints into classical flow constraints (like Chapters 2, 3 and 4) and practical problems with real data and technical constraints related to electricity (Chapter 5). We study problems which are generalizations of the Steiner tree problem: given a graph with a set of vertices, a set of edges and a subset of vertices called terminals, this problem aims at finding a tree of minimum cost spanning all the terminals. The vertices which are not terminals are called Steiner vertices. We introduce a root vertex in our problems. In our wind-farm application, the wind turbines correspond to the terminals, the substation to the root, and the inter-connection nodes to the Steiner vertices. We aim at solving generalizations of Steiner tree problems taking robustness and edge capacities into account.

In Chapter 1.2, we introduce and define some notions, notations and methods used in this thesis. We also define the studied problematics. We present some preliminary results and summarize some previous works found in the literature.

Following discussions with EDF engineers, it appeared that the electrical constraints can be formulated as classical flow constraints if the solution network is an arborescence. Hence, in Chapter 2, we reduce the problem to the search for a Steiner arborescence which respects the capacity constraints. We focus on the design of Steiner arborescences for which we try to reduce the damaging impact of an arc deletion in the arborescence solution, according to several optimization criteria, at a reasonable cost. We give complexity results and propose several formulations tested on real data.

In our wind farm applications, the terminals produce energy, so the energy flow should be routed from the terminals to the root. However, it is equivalent to consider that we route the energy from the root to the terminals (in a digraph, we just have to take the opposite arcs whereas there are no changes in an undirected graph). Furthermore, in offshore wind farm applications, we often consider that the energy produced by the wind turbines is uniform, which is equivalent to consider a unit demand and an arc capacity can then be seen as the number of terminals that can be linked to the root through this arc.

In Chapter 3, we define the Capacitated Rooted k-Edge Connected Steiner Network: given a connected graph with a root vertex, a set of terminals and integer capacity and cost on each arc of the graph, we aim to find a subset of arcs of minimum cost such that there exists a feasible flow (i.e. respecting the capacities) routing one unit of flow from the root

to each terminal in the subgraph induced by those arcs, even if a given number k of arcs is deleted. This problem is equivalent to finding a robust cabling network in a wind farm, when the electricity constraints are formulated as classical flow constraints. We give several formulations, including a new bilevel formulation, study the relations between the different formulations, and test them in order to compare their efficiency.

In Chapter 4, we focus on planar graphs for the Capacitated Rooted k-Edge Connected Steiner Network. We present a method to check whether a network is resilient or not to the deletion of any set of k arcs using planar graph duality and shortest paths problems. We propose and describe a tabu search algorithm derived from this method. We test our algorithm and compare its efficiency to exact methods presented in Chapter 3.

In Chapter 5, we study the real-life problem of designing a wind farm cabling network with electrical constraints of load flow. We explain the *load flow* study, which corresponds to a numerical analysis of the flow of electric power in an interconnected system. We must ensure that the power routed through each cable respects the electric capacities of the cable using the load flow equations to analyze the state of the electric network once the network is built. Since the load flow analysis is a non-linear system, we use an approximation in order to include them into a mixed-integer linear program. We test our algorithm on real data and give numerical results.

In a concluding chapter, we give some perspectives and future work leads.

1.2 Preliminaries

1.2.1 General notions in graph theory

In this section, we recall some notions from graph theory that will be used in this thesis. For more information about graph theory, the reader is referred to [12, 63].

Formally, a directed graph (or digraph) $G = (V, A)$ is defined by a set of vertices V and a set of arcs $A \subseteq V \times V$. The set of predecessors (respectively successors) of a vertex v in G is defined by $\Gamma^-_G(v)$ (respectively $\Gamma^+_G(v)$). For a subset of vertices $S \subset V$ in G, we define $\delta^-_G(S) = \{(i,j) \in A \mid i \in V \setminus S,\ j \in S\}$ (respectively $\delta^+_G(S) = \{(i,j) \in A \mid i \in S,\ j \in V \setminus S\}$) as the set of arcs entering (resp. leaving) S. When there are no ambiguities about the related graph G, we can refer to those sets as $\Gamma^-(v)$, $\Gamma^+(v)$, $\delta^-(S)$ and $\delta^+(S)$, respectively.

An undirected graph $G' = (V, E)$ is defined by a set of vertices V and a set of edges $E \subseteq V \times V$. The set of neighbors of a vertex $v \in V$ is denoted by $\Gamma_{G'}(v)$. For a subset of vertices $S \subset V$ in G', we define $\delta_{G'}(S)$ as the set of edges $[u, v]$ incident to S in G' ($u \in S$ and $v \in V \setminus S$). When there are no ambiguities about the related graph G', we can refer to those sets as $\Gamma(v)$ and $\delta(S)$ respectively.

Two paths p_1 and p_2 in a graph are said to be arc-disjoint (edge-disjoint in an undirected graph) if there is no arc a which appears both in p_1 and p_2. In this thesis, we will consider a special vertex r of a graph G, which is the root of the graph (i.e. there exists a path between r and every vertex of G). A graph is said to be connected if there is an undirected path between u and v for each pair of vertices $u, v \in V^2$. In a digraph $G = (V, A)$ (resp. an undirected graph $G = (V, E)$), an $(i, j)-$disconnecting set, with i and j two distinct vertices of V, is a set of arcs $D \subseteq A$ (resp. a set of edges $D \subseteq E$) such that there are no path from i to j in the graph $G' = (V, A \setminus D)$ (resp. $G' = (V, E \setminus D)$). A graph (respectively digraph) is said to be k-edge-connected (resp. k-arc-connected) if it remains connected (resp. strongly connected) whenever fewer than k edges (resp. arcs) are removed.

Theorem 1.2.1 (k-edge-connectivity [48]) *If i and j are two vertices of a digraph (resp. undirected graph), the minimum size of an $(i, j)-$disconnecting set is equal to the maximum number of pairwise arc-disjoint (resp. edge-disjoint) paths from i to j. Moreover, a graph G is k-edge-connected if and only if there exist k pairwise arc-disjoint paths (resp. edge-disjoint) between each pair of vertices in G.*

1.2.2 Flows and networks

We introduce here different notions about network flows; the reader is referred to [2] for more information about this topic. We define a network as a digraph $G = (V, A)$ with a non-negative capacity u_{ij} on each arc $(i, j) \in A$, a distinguished root $r \in V$ (also called source vertex) and a sink vertex $s \in V$. A flow x is a function that assigns a value x_{ij} to each arc $(i, j) \in A$. A feasible flow satisfies the capacity constraints $0 \leq x_{ij} \leq u_{ij}$ for each arc $(i, j) \in A$ and the conservation constraints $\sum_{u \in \Gamma^-(v)} x_{uv} = \sum_{w \in \Gamma^+(v)} x_{vw}$ for each vertex $v \in V \setminus \{r, s\}$. The value of such a flow is equal to $\sum_{v \in \Gamma^+(r)} x_{rv} - \sum_{u \in \Gamma^-(r)} x_{ur} = \sum_{i \in \Gamma^-(s)} x_{is} - \sum_{j \in \Gamma^+(s)} x_{sj}$. In this thesis, we assume without loss of generality that, in any network we consider, there are no arcs entering the root r and no arcs leaving the sink s. Thus, the value of a flow

x is equal to $\sum_{v\in\Gamma^+(r)} x_{rv} = \sum_{i\in\Gamma^-(s)} x_{is}$. A maximum flow is defined as a feasible flow of maximum value.

In a network, for a given subset of vertices $S \subset V$ with $r \in S$ and $s \in V \setminus S$, we refer to the partition $[S, V \setminus S]$ of V as an $r-s$ cut. The cut-set associated with S is the set of arcs going from a vertex of S to a vertex of $V \setminus S$, i.e. the cut-set corresponds to $\delta^+(S)$ or $\delta^-(V \setminus S)$. The capacity of a cut (or the capacity of a cut-set) is defined as the sum of the capacities of the arcs of its cut-set, i.e. $\sum_{(i,j)\in\delta^+(S)} u_{ij}$. We recall the well-known following theorem:

Theorem 1.2.2 (Max-flow min-cut theorem [29]) *In a network with a root r and a sink s, the minimum capacity of a $r-s$ cut is equal to the maximum value of a feasible flow from r to s.*

The problem of searching for a maximum flow (respectively a minimum $r-s$ cut) in a network is called the *maximum flow problem* (respectively the *minimum cut problem*). The flow x_{ij} on each arc $(i,j) \in A$ is not constrained to be integer. However, in this thesis we will work on integer flows because of the applications we are considering. Thus, we remind the following theorem:

Theorem 1.2.3 (Integrality Theorem [2]) *If all capacities in a network are integer, then there exists a maximum flow in which the amount of flow on each arc is integer. Furthermore, a maximum flow can be partitioned into flows of integer values along paths from the root to the sink.*

Theorem 1.2.3 allows to relax the integrality constraint on the value of the flow when searching for a maximum flow. We introduce the following well-known linear formulation for the maximum flow problem:

$$(MAX-FLOW) \left| \begin{array}{lll} \max\limits_{x} \sum\limits_{v\in\Gamma^+(r)} x_{rv} & & \\ \text{s.t.} \sum\limits_{i\in\Gamma^-(j)} x_{ij} - \sum\limits_{k\in\Gamma^+(j)} x_{jk} = 0 & \forall j \in V\setminus\{r,s\} & (1.1a) \\ 0 \leq x_{ij} \leq u_{ij} & \forall (i,j) \in A & (1.1b) \end{array} \right.$$

where the variable x_{ij} defines for each arc (i,j) the amount of flow routed through (i,j). We introduce Remark 1.2.1:

Remark 1.2.1 *If there exist two vertices i and j in $V \setminus \{r, s\}$ such that $\{(i, j), (j, i)\} \subset A$ and a flow defined by x such that $x_{ij} > 0$ and $x_{ji} > 0$, then there always exists a flow x' of equal value with either $x'_{ij} = 0$ or $x'_{ji} = 0$.*

Proof: Let us suppose that a given feasible flow x assigned to a network $G = (V, A)$ is such that there exist two vertices i and j in V such that $(i, j), (j, i) \in A$ and there is a positive amount of flow on both arcs (i.e. $x_{ij}x_{ji} > 0$). By reducing the flow on both arcs by $min(x_{ij}, x_{ji})$, we find a flow x' with either $x'_{ij} = 0$ or $x'_{ji} = 0$. The capacity constraints are obviously satisfied because we only reduce the amount of flow on two arcs and they were satisfied by x. The conservation constraints are also satisfied since we reduce the amount of flow entering and leaving both i and j by $min(x_{ij}, x_{ji})$. Finally, the value of the flow defined by x remains the same because $\sum_{i \in \Gamma^-(s)} x_{is}$ is not changed. □

A matrix A is said to be *totally unimodular* if each square submatrix of A has a determinant equal to -1, 0 or 1 (see [58]). In particular, we have that each entry of A is -1, 0 or 1.

Theorem 1.2.4 (Theorem 5.20 in [58]) *Let A be a totally unimodular $m \times n$ matrix and let $b \in \mathbb{Z}^m$. Then the polyhedron*

$$P = \{x | Ax \leq b\}$$

is integer.

We have the following property and theorem on the totally unimodular matrices:

Property 1.2.1 *Let A be a totally unimodular $m \times n$ matrix and I^m the $m \times m$ identity matrix. We have that $-A$, $A^\top$ and $[A|I^m]$ are totally unimodular matrices.*

The matrix $\mathcal{M}$ associated with the left-hand side of Constraints (1.1a) is a sub-matrix of the *node-arc incidence matrix* (we remove the rows associated to r and s) of a digraph. Each column of the matrix corresponds to an arc (i, j): there is a 1 in the ith row and a -1 in the jth row and the rest of the entries of the column are 0.

Theorem 1.2.5 (Theorem 11.12 in [2]) *The node-arc incidence matrix $\mathcal{M}$ of a directed network is totally unimodular.*

Following Theorems 1.2.4 and 1.2.5, we do not have to ensure that x is integer since $\mathcal{M}$ is totally unimodular, which yields a proof of Theorem 1.2.3.

The dual of $(MAX - FLOW)$ corresponds to a formulation of the minimum-cut problem and can be written as follows:

$$
\begin{array}{llll}
\min\limits_{\mu,\lambda} & \sum\limits_{(i,j)\in A} u_{ij}\lambda_{ij} & & \\
\text{s.t.} & \mu_v + \lambda_{rv} \ \geq \ 1 & \forall v \in \Gamma^+(r) & (1.2a) \\
& \mu_v - \mu_u + \lambda_{uv} \ \geq \ 0 & \forall (u,v) \in A, u \neq r, v \neq s & (1.2b) \\
& -\mu_u + \lambda_{us} \ \geq \ 0 & \forall u \in \Gamma^-(s) & (1.2c) \\
& \lambda_{ij} \geq 0 \quad \forall (i,j) \in A, \quad \mu_v \in \mathbb{R} \quad \forall v \in V \setminus \{r,s\} & &
\end{array}
$$

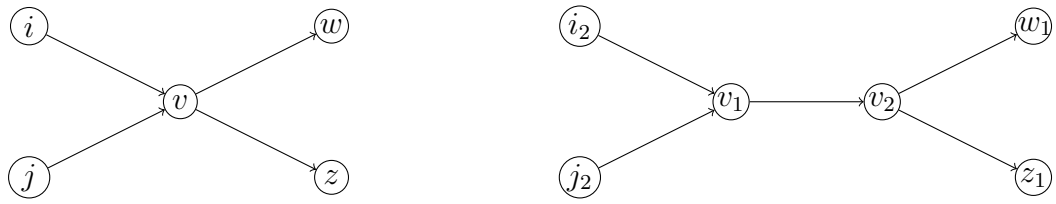
It can be reformulated, with the addition of variables μ_r and μ_s, as follows:

$$
(MIN - CUT) \left|
\begin{array}{llll}
\min\limits_{\mu,\lambda} & \sum\limits_{(i,j)\in A} u_{ij}\lambda_{ij} & & \\
\text{s.t.} & \mu_v - \mu_u + \lambda_{uv} \ \geq \ 0 & \forall (u,v) \in A & (1.3a) \\
& \mu_r \ = \ 1 & & (1.3b) \\
& \mu_s \ = \ 0 & & (1.3c) \\
& \lambda_{ij} \geq 0 \quad \forall (i,j) \in A, \quad \mu_v \in \mathbb{R} \quad \forall v \in V & &
\end{array}
\right.
$$

In any optimal solution we have $\mu_v \in [0,1] \quad \forall v \in V$, which implies $\lambda_{ij} \leq 1 \quad \forall (i,j) \in A$. From Property 1.2.1, the transpose of a totally unimodular matrix is totally unimodular, so the matrix of constraints of $(MIN - CUT)$ is totally unimodular, and thus there exists an optimal solution (λ^*, μ^*) with $\mu^* \in \{0,1\}^{|V|}$ and $\lambda^* \in \{0,1\}^{|A|}$. The dual problem then formulates the minimum cut problem in this way: μ defines the partition associated with the cut (we have $\mu_v = 1$ if v is in the same part as the root and $\mu_v = 0$ if v is in the same part as the sink) while λ defines the cut-set (we have $\lambda_{ij} = 1$ if the arc (i,j) is in the cutset, i.e. $\mu_i = 1$ and $\mu_j = 0$).

In this thesis, we will consider arc-deletions in flow networks. However, vertices-deletion can be considered using the same methods by a simple transformation of the input graph. The graph transformation is the following: we replace each vertex v of the input graph G by two vertices v_1 and v_2 and an arc (v_1, v_2) in the transformed graph and each arc (u,v) is replaced by an arc (u_2, v_1). The deletion of the vertex v in the input graph then corresponds to the deletion of the arc (v_1, v_2) in the transformed graph (as illustrated in the example in Figure 1.1). For the non-oriented case, we can apply the transformation to the bi-directed graph

associated to the input graph.



(a) The vertex v in the input graph (b) The arc (v_1, v_2) in the transformed graph

Figure 1.1 Example of graph transformation for the vertex v

1.2.3 Steiner trees and networks

We introduce in this subsection the concepts of Steiner trees and Steiner networks (see [25, 40, 52]). A *tree* is defined as a connected graph which is acyclic. Given an undirected graph $G = (V, E)$ and a subset of vertices $T \subseteq V$, a subgraph S of G is called a *Steiner tree* if S is a tree containing all vertices of T. A vertex $t \in T$ is called a *terminal vertex* (or *terminal*) whereas a vertex $v \in V \setminus T$ is called a *Steiner vertex* (or *Steiner point*). The minimum Steiner tree problem can be defined as follows:

Minimum Steiner Tree Problem

INSTANCE: A graph $G = (V, E)$ and a set of terminals $T \subseteq V$.

PROBLEM: Find a minimum Steiner tree spanning T in G. That is, find a Steiner tree $S = (V_S, E_S)$ such that $|E_S| = min\{|E_{S'}| \mid S' = (V_{S'}, E_{S'})$ is a Steiner tree spanning T in G$\}$.

Given a positive cost c_e for each edge $e \in E$, we define similarly the **Minimum-Cost Steiner Tree Problem** as the problem of finding a Steiner tree $S = (V_S, E_S)$ for which the sum of the costs $c_{e'}$ for $e' \in E_S$ is minimum. The Minimum-Cost Steiner Tree problem is $\mathcal{NP}$-hard and generalizes both the Minimum Spanning Tree problem and the Shortest Path problem: if $T = V$, the Minimum Steiner Tree Problem is equivalent to the Minimum Spanning Tree Problem whereas if $|T| = 2$, it is equivalent to the Shortest Path problem.

In this thesis, we consider Steiner problems with a root and a capacity u_{ij} on each edge $[i, j]$ in the graph. We consider that each terminal $t \in T$ has the same demand and we want to find a feasible flow (i.e. respecting the capacities defined by u and the flow conservation

constraints) such that a unit of flow is routed from the root to each terminal: each terminal can hence be seen as a sink. We say that a feasible flow x routes one unit of flow from the root r to each terminal if the flow conservation constraints are as follows:

$$\sum_{k \in \Gamma^+(j)} x_{jk} - \sum_{i \in \Gamma^-(j)} x_{ij} = \begin{cases} |T| & \text{if } j = r \\ -1 & \text{if } j \in T \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in V$$

If we add to the graph a fictive sink s with an edge (t, s) of capacity $u_{ts} = 1$ for each terminal $t \in T$, our problem is equivalent to finding a feasible $(r - s)$-flow of value $|T|$.

1.3 Previous work

We summarize in this section the existing results in the literature for problems which are related to the ones studied in this thesis.

1.3.1 Wind farm cable layout optimization

The design of wind farms brings several challenges in optimization: one can think of the optimization of the location of each wind turbine (a literature review for this kind of problems is proposed by Gonzalez et al. [37]) or of the connection between the electric network and the wind farm [53], for example. In this thesis, we will only consider the problem of designing the cabling network that collects the energy produced by the wind turbines and route it to the sub-station, once the locations of the different wind turbines are known.

Problems of designing the wind farm cabling network have already been studied in the literature. Hertz et al. [39] study a real-life problem with real data for onshore wind farms, where several cable types are available (subterranean or not). Furthermore, the energy produced by the wind turbines and routed to the sub-station is non-splittable: once the energy is routed through a same cable, it cannot be split and must be routed to the sub-station through the same path (i.e. if a "chunk" of energy $\mathcal{E}$ is routed from u to v through the same cable with v different from the substation, there exists a node w such that the "chunk" $\mathcal{E}$ is entirely routed from v to w). The authors give mixed integer formulations and propose a cutting plane generation method allowing to evaluate their algorithm on real data.

Other authors propose mixed integer formulations for the design of wind farm cabling [14, 27]. Regarding offshore wind farms, Pillai et al. [51] propose a set of algorithms computing ap-

proximate solutions in order to optimize successively the location of the wind turbines and the design of the wind farm cabling network considering natural obstacles related to the environment. Fischetti and Pisinger [28] also consider natural obstacles related to the environment but they additionally consider the power losses related to the routing of electricity. They propose a mixed-integer formulation and a method allowing to find good approximate solutions using matheuristics.

To the best of our knowledge, there does not exist literature proposing a method that could be applied to the problem of designing wind farm cabling networks with resilience to breakdowns on cables and load flow constraints. In this thesis, we are interested in the design of wind farm networks with such constraints.

1.3.2 Steiner arborescence problems

In some cases, production rules and electrical constraints related to the routing of electricity imply that the problem of designing such a wind farm cabling can be seen as the search for a Steiner arborescence with capacity constraints and unitary demands. More precisely, the problem can be defined as follows: we are given a graph with a subset of vertices T and a root r, where each arc has a cost and a capacity. We look for an arborescence rooted at r which contains each vertex of T, and such that, for each arc (i, j) of the arborescence, the number of terminals in the sub-arborescence rooted at j is at most equal to the capacity of (i, j). The problem is then a generalization of the Steiner arborescence problem, but also a particular case of the generalized Steiner arborescence, in which there is a fixed demand at each terminal and each arc capacity corresponds to the quantity of demands which can be routed through this arc.

The minimum-cost (or weighted) Steiner tree problem (without capacity constraints) has been widely studied in the literature; the reader is referred to [25, 40, 52]. It has also many applications in industry; see [22, 26]. This problem is $\mathcal{NP}$-hard even if all edge costs are equal (it corresponds to the **Minimum Steiner Tree Problem**) and if G is planar [32]. However, if the number of terminals is fixed, this problem can be solved in polynomial time [24].

When taking into account the arc capacities, Papadimitriou shows that the problem is $\mathcal{NP}$-Hard in the spanning case (i.e. all vertices except the root are terminals) [49]. The problem has been studied in the spanning case, and branch-and-bound as well as branch-and-price

algorithms have been proposed [21, 62]. Jothi and Raghavachari [41] and Arkin et al. [4] propose approximation algorithms when capacities are uniform.

Regarding Steiner arborescence problems with capacity constraints, Bentz et al. study the complexity and approximation considering several parameters like the number of terminals, the arc costs and the capacities [10]. Goemans and Myung propose several Steiner tree formulations [36]. Bousba et al. solve the Steiner arborescence problem with capacities and demands, which is a generalization of the Steiner tree problem with capacities [20]: each terminal has a specific amount of demand that must be routed from the root to this terminal, and the capacity of an arc corresponds to the maximum amount of demands that can be routed through this arc.

To the best of our knowledge, the design of Steiner trees with constraints of robustness, i.e. where we aim to design trees which are not too much impacted by arc deletions, has not been studied. However, different problems with arc deletions on trees have been studied. Bazgan et al. [8] study the problem of finding in a graph a subset of k edges whose deletion causes the largest increase in the weight of a minimum spanning tree: they propose an enumeration algorithm and a MIP to solve the problem. This problem has been shown to be $\mathcal{NP}$-hard, and approximation algorithms have been proposed [30, 45]. However, this problem considers the arc deletions before the design of the tree, whereas we consider the arc deletions during the design of the tree.

1.3.3 Robust Steiner networks

Robust problems have been widely studied in continuous optimization [9, 15, 43] and can be seen as problems modeling uncertainty, where the description of uncertainty is a deterministic variability of data or parameters. In this thesis, we consider an uncertainty on the arcs (or cables in our application on wind farms): an arc can be deleted or not. The robust aspect of "worst-case minimization" that we consider can be seen as the one proposed by Bertsimas and Sim [13], who set an upper bound on the total data uncertainty (budget of uncertainty).

More precisely, we address problems of designing networks in which we consider the possibility of arc deletion: in our application, it corresponds to taking into account the risk of a breakdown on one or several cables after the construction of the wind farm cabling network. We estimate a budget of arc deletions k: there can be at most k simultaneous arc deletions in the network (i.e. k breakdowns at the same time). The problem is then to design a network

which is still functional after the deletion of any set of k arcs.

In the literature, problems of designing *survivable* networks have been studied; the term has been introduced by Steiglitz et al. [60]. Given a graph $G = (V, E)$ and a cost function on E, the problem consists in finding a subgraph of G of minimum cost which respects some connectivity constraints. However, these connectivity constraints can be defined in several ways in the literature.

On the one hand, it is possible to define a matrix $R = [r_{ij}]$: a feasible solution must then contain r_{ij} disjoint paths between each pair of vertices i and j. On the other hand, connectivity requirements can be defined by a connectivity value r_v given for each vertex v, and we have to ensure that we have $\min(r_i, r_j)$ disjoint paths for each pair of vertices i and j. In order to well dissociate the two problems, we refer to the first one as the Network Design Problem with Connectivity Requirements (**NDC**) [47] and to the second one as the Survivable Network Problem (**SNP**) [60]. SNP is trivially a special case of NDC. Furthermore, the definition of both problems can vary if we consider vertex-disjoint or edge-disjoint (or arc-disjoint in directed graphs) paths. Those problems generalize well-studied problems such as the minimum spanning tree (when all requirements are equal to 1), the minimum Steiner tree (when all requirements are equal to either 1 or 0) or the design of k-connected graphs at minimum cost (when all requirements are equal to 0 or a given integer k).

Goemans and Bertsimas consider SNP in the case of edge-disjoint paths when the input graph is complete [35]. They give problem formulations and properties on the structure of the continuous relaxation. They also propose a heuristic which is based on solving several Steiner tree problems. This heuristic ensures a cost value of at most $2\min(\log R, p)$ times the cost of an optimal solution, where R is the highest connectivity requirement and p is the number of non-zero values in the connectivity requirements.

Raghavan [54] proposes a dual-ascent algorithm and new formulations for NDC in the case of both vertex-disjoint and edge-disjoint paths. Williamson et al. [64] give an approximation algorithm running in polynomial time with an approximation radio equal to $2R$ (where R is defined as previously), which has been enhanced afterward by Gabow et al. who propose an algorithm with a better time complexity [31]. Agrawal et al. [1] propose an approximation algorithm for NDC in the case of arc-disjoint paths with a ratio equal to $2\log R$ when considering that an arc of the input graph can be selected several times in a solution.

Grötschel et al. [38] give properties on the structure of an optimal solution for NDC allowing to design efficient heuristics. They also study the polyhedral structure of the problem and propose a cutting plane algorithm based on those results.

Mixed Integer Linear Programs for solving those problems often consist in formulations based on the cut-sets of the input graphs. Magnanti and Raghavan [47] propose a formulation for NDC in the case of edge-disjoint paths based on multi-commodity flows. They show that this formulation is stronger than the one using cut-set. A method based on Benders decomposition is proposed by Botton et al. for a problem with hop-constraints (the lengths of the different paths between vertices with connectivity requirements should not exceed a given parameter) [19]. Kerivin and Mahjoub give a survey on this type of problems [42].

One of the main problems we study is called the Capacitated Rooted k-Edge Connected Steiner Network Problem: we aim to design a network of minimum cost in which, after the deletion of any set of k arcs, we can still find a feasible flow (i.e. a flow respecting the capacities) routing one unit of flow from the root to each terminal. Grotschel et al. study this problem but do not take into account capacities, and use the connectivity requirement function r like in SNP [38]. In our case, the connectivity requirement can be seen as using the requirement matrix $R = [r_{ij}]$ as in the case of NDC, with the restriction that for each terminal t, we have $r_{rt} = k + 1$ (where r denotes the root vertex), and the value of each other entry in R is 0. Several authors take into account capacities by considering a cost of allocation, whereas we consider fixed capacities [16, 55]. Studies on more generalized problems with multi-commodity flows have also been considered [23, 61]. There also exist studies of those problems in particular graphs without the existence of root or the capacity constraints [6, 17]. In this thesis we study the design of networks resilient to a given number of arc-deletions and a single-source defined by the root, while taking fixed capacities into account, whereas in the literature capacity allocation has been studied.

1.3.4 The k most vital arcs in flow networks and network interdiction problems

Given a network with arc capacities, a root (or source) vertex r and a sink vertex s, the problem of finding a subset of k arcs such that the deletion of these k arcs results in the maximum decrease of the value of the maximum flow between r and s can be referred as the k Most Vital Arcs Problem in Flow Networks (k-**MVAPFN**).

Lubore et al. [46] and Wollmer [65] give fast algorithms for the case where $k = 1$ using a sequence of maximum flow problems. Barton [7] proposes more efficient algorithms to solve the same problem for particular graphs like acyclic graphs.

Ratliff et al. [56] introduce the problem for a non-fixed value of k. The problem has been generalized into the network interdiction problem by Wood [66]: he considers a deletion cost associated with each arc and a budget B of deletion, the sum of the deletion costs of the arcs which are deleted must not exceed the budget B. Obviously, k-MVAPFN is a special case of the network interdiction where $B = k$ and all deletion costs are equal to 1. He shows that the problem is strongly $\mathcal{NP}$-hard for both k-MVAPFN and the network interdiction problem, and proposes a mixed-integer formulation. This problem has several applications [5, 33, 57].

On planar graphs, the problem becomes weakly $\mathcal{NP}$-complete: Phillips [50] and Zenklusen [68] propose algorithms using planar graph duality to solve the problem in pseudo-polynomial time. We present in this thesis a tabu search using an extension of those results in order to check whether a solution is feasible or not.

CHAPTER 2 ROBUST ARBORESCENCES

2.1 Introduction

In this chapter, we focus on finding a robust Steiner or spanning arborescence covering the root and the terminals of G. Here, the robustness consists in finding a solution which minimizes the number of terminals disconnected from the root in the worst case of an arc failure.

This setting arises in some wind farm cabling problems, when technical constraints impose that all electrical flows arriving at any device except the substation must leave it through one and only one cable: an inclusion-wise minimal sub-network of G respecting those constraints then corresponds to a Steiner anti-arborescence. The wind turbines are identical, and the wind is assumed to blow uniformly, so we can assume without loss of generality that each turbine produces one unit of energy. Then, A is the set of all possible cable locations, r is the sub-station collecting the energy and delivering it to the electric distribution network, T represents the set of nodes where a wind turbine lies, and $V \setminus (\{r\} \cup T)$ is the set of Steiner nodes, corresponding to possible junction nodes between cables. In that case, the flow is routed from the vertices of T to r, and we search for an anti-arborescence. However, the problem is easily seen to be equivalent to the Steiner arborescence problem, by reversing the flow circulation in the solution.

We begin by defining the problem and giving some complexity results, and then we propose mathematical formulations which are tested on real wind farm instances.

2.2 Definition of problems and complexity results

We assume in this section that the graph $G = (V, E)$ is undirected. We define the robust problem without capacity constraints as follows:

Robust Steiner Arborescence problem (RStA)

INSTANCE: A connected graph $G = (V, E, r, T)$ with $r \in V$ and $T \subseteq V \setminus \{r\}$.

PROBLEM: Find an arborescence $S = (V_S, A_S)$ such that $V_S \subseteq V$, $A_S \subseteq E$ and $T \subset V_S$,

which is rooted at r and minimizes the number of terminals disconnected from r when an arc a is removed from A_S, in the worst case.

We also consider the spanning version of the problem (i.e., $T = V \setminus \{r\}$). In this case, the problem is to minimize the number of vertices in the largest (regarding the number of vertices) subarborescence not containing r. We define it as follows:

Robust Spanning Arborescence problem (RSpA)

INSTANCE: A connected graph $G = (V, E, r)$ with $r \in V$.

PROBLEM: Find a spanning arborescence S of G, rooted at r, which minimizes the size of the largest subarborescence of S not containing r.

Obviously, the largest subarborescence not containing r is rooted at a vertex $v \in \Gamma_G(r)$, and the worst case is the failure of an arc incident to the root. We have the following property:

Property 2.2.1 *a) There is an optimal solution S^* of RSpA containing (r, v) for all $v \in \Gamma_G(r)$ ($\Gamma_G(r) = \Gamma^+_{S^*}(r)$).*

b) There is an optimal solution $S^* = (V^*, A^*)$ of RStA containing (r, v) for all $v \in V^* \cap \Gamma_G(r)$.

Proof: Let $S = (V, A_S)$ be an optimal solution of RSpA such that there is $v \in \Gamma_G(r)$ with $(r, v) \notin A_S$, and let w be the predecessor of v in the path from r to v in S. If we remove (w, v) from A_S and add (r, v), we obtain a new spanning arborescence at least as good as S, since we have replaced a subarborescence by two subarborescences of smaller sizes. Doing so for each $v \in \Gamma_G(r)$ with $(r, v) \notin A_S$ yields a solution S^* verifying the property.

The proof is similar for RStA, by replacing $\Gamma_G(r)$ by $V^*_S \cap \Gamma_G(r)$: if we remove (w, v) from A_S and add (r, v), we obtain a new Steiner arborescence at least as good as S, since we have replaced a subarborescence by two subarborescences spanning at most the same number of terminals. □

Notice that the property does not hold if we have capacity constraints, because the capacity of (r, v) can be smaller than the one of (w, v) in the proof above. Let us now introduce the feasibility problem associated with **RSpA**:

Robust Spanning Arborescence Feasibility problem (RSpAF)

INSTANCE: A connected graph $G = (V, E, r)$ with $r \in V$ and an integer β with $1 \leq \beta \leq |V| - 1$.

QUESTION: Is there a spanning arborescence $S = (V_S, A_S)$ of G, rooted at r, such that the size of any subarborescence of S not containing r is at most β?

Theorem 2.2.1 ***RSpAF*** *is NP-Complete.*

Proof: We introduce the 3-Partition problem [32] in order to transform an instance of this problem into a **RSpAF** one.

3-Partition problem

INSTANCE: A finite set D of $3m$ positive integers d_i, $i = 1, .., 3m$, and a positive integer B such that $\sum_{i=1,...,3m} d_i = mB$ and $B/4 < d_i < B/2$ $\forall i = 1, ..., 3m$.

QUESTION: Can D be partitioned into m disjoints subsets $M_1, M_2, ..., M_m$ of three elements such that the sum of the numbers in each subset is equal to B?

To obtain an instance of **RSpAF** from an instance of 3-Partition, we set $\beta = B + 1$ and we construct the following graph $G = (V, E)$: we define a root r and m vertices v_j with an edge $[r, v_j]$ for $j = 1, ..., m$, each vertex v_j corresponding to a set M_j. We add $3m$ vertices w_i and the edges $[v_j, w_i]$ for all $j = 1, .., m$ and all $i = 1, .., 3m$, each vertex w_i corresponding to the element d_i of D (the subgraph induced by the vertices v_j and w_i is complete bipartite). Finally, for each $i = 1, .., 3m$, we add $d_i - 1$ vertices adjacent to w_i : the subgraph induced by those vertices and the vertices w_i is made of $3m$ disjoint stars. See Figure 2.1 for a graph representation of a 3-Partition instance with $m = 2$, $B = 11$ and $D = \{5, 3, 4, 3, 4, 3\}$. Notice that $|V| = 1 + m + mB$.

Solving **RSpAF** on G with $\beta = B + 1$ amounts to finding an arborescence where the size of the subarborescence rooted at each v_j is smaller than or equal to $B + 1$. If there is a solution to **RSpAF** on G, then, from the proof of Property 2.2.1, there is a solution S such that $(r, v_j) \in S$ $\forall j = 1, ..., m$, and each w_i is connected to exactly one v_j, otherwise there is a cycle. Given a vertex $v \in S$, let $S(v)$ be the subarborescence of S rooted at v: $\forall j = 1, ..., m$, we have $|S(v_j)| \leq B + 1$ and $\sum_{j=1,..,m} |S(v_j)| = |V \setminus \{r\}| = mB + m$. Thus, $\forall j = 1, ..., m$, $|S(v_j)| = B + 1$ and $S(v_j)$ contains v_j and several vertices w_i, each having $d_i - 1$ successors in S. Finally, the constraints $B/4 < d_i < B/2$ imply that, $\forall j = 1, ..., m$, v_j is connected to exactly 3 vertices w_i denoted in the following by w_{j_1}, w_{j_2} and w_{j_3}, and such

that $|S(w_{j_1})| + |S(w_{j_2})| + |S(w_{j_3})| = |S(v_j)| - 1 = B$.

Then, it is easy to obtain a solution to the 3-Partition instance. For each $j = 1, .., m$, we set $M_j = \{|S(w_{j_1})|, |S(w_{j_2})|, |S(w_{j_3})|\} = \{d_{j_1}, d_{j_2}, d_{j_3}\}$. We have m disjoint sets, each of size B, which cover exactly D. For the instance given in Figure 2.1, a solution to 3-Partition can be associated with the arborescence given in thick: $M_1 = \{5, 3, 3\}$ and $M_2 = \{4, 4, 3\}$.

Moreover, from a solution to the 3-Partition instance, it is easy to obtain a solution S to **RSpAF** for the associated graph G, using similar arguments.

The 3-Partition problem is NP-Complete in the strong sense, meaning that it remains NP-Complete even if the integers in D are bounded above by a polynomial in m. Thus, the reduction can be done in polynomial time and **RSpAF**, which is clearly in NP, is NP-Complete. □

RSpAF being NP-Complete, **RSpA** is NP-Hard, and so is **RCStA** because it is a generalization of **RSpA**. Let us now consider capacity constraints on the edges. **RSpAF** can be seen as a special case of the general capacitated spanning arborescence problem where the demand at each node is an integer (our demands are all equal to 1), and hence from Theorem 2.2.1 we obtain the following corollary:

Corollary 2.2.2 *Given a graph $G = (V, E, r, d, u)$ where d represents the (integral) demands at each node and u the capacities of the edges, the problem of deciding whether there exists a spanning arborescence of G, rooted at r and respecting the capacities, is NP-Complete (even if u is a uniform function and all demands are equal to 1).*

This extends the following result due to Papadimitriou [49]: given two positive values C and K and a graph $G = (V, E, r, c)$ where c is a cost function on the edges, the problem of deciding whether there exists a spanning arborescence S of G rooted at r, such that each subarborescence of S not containing r contains at most K vertices, and with total cost at most C, is NP-Complete.

The complexity results given in this section concern undirected graphs, and so the more general case of directed graphs too, since an undirected graph can be transformed into a directed one by replacing each edge by two opposite arcs. If we consider problems with capacity constraints, we give the same capacity to both opposite arcs: since we search for an

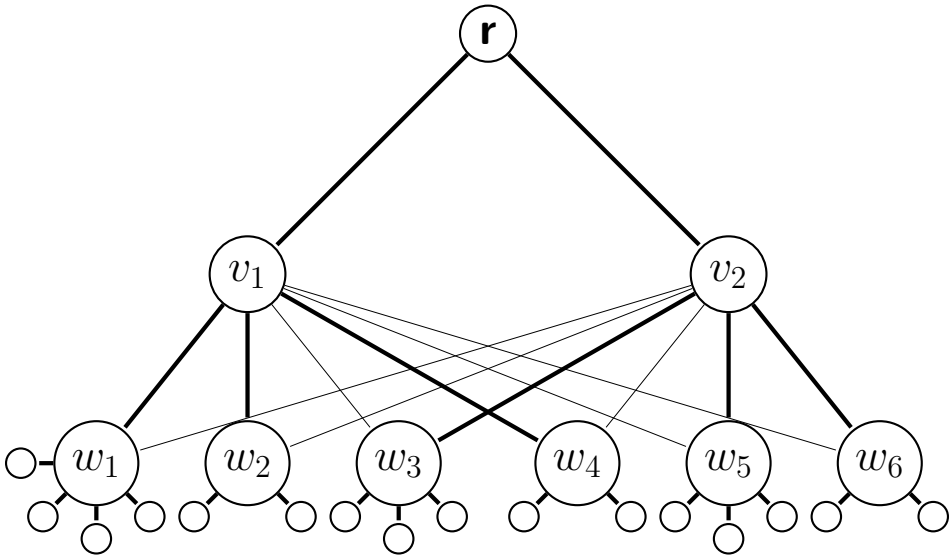


Figure 2.1 Graph and **RSpAF** solution resulting from the 3-Partition instance in which $m = 2$, $B = 11$, $D = \{5, 3, 4, 3, 4, 3\}$

arborescence, only one of them will appear in the solution.

In the following, we study the more general following problem, which is hence also NP-hard:

Robust Capacitated Steiner Arborescence problem (RCStA)

INSTANCE: A connected graph $G = (V, E, r, T, u)$ with $r \in V$, $T \subseteq V \setminus \{r\}$ and u a positive integer function on E.

PROBLEM: Find an arborescence $S = (V_S, A_S)$ with $V_S \subseteq V$ and $A_S \subseteq E$, rooted at r and spanning the terminals of T, which respects the arc capacities and minimizes the number of terminals disconnected from r when an arc a is removed from A_S, in the worst case.

2.3 Mathematical formulations and tests

In this section we propose formulations for robust Steiner problems where the robustness is considered either as a constraint with the objective of minimizing the cost, or as an objective with or without constraints on the cost. Moreover, we study two kinds of robustness by considering worst or average consequences of breakdowns.

Let $G = (V, A, r, T, u, c)$ be a directed graph with a vertex set V, an arc set A, a root r, a

set of terminals T, and capacity and cost functions, respectively denoted by u and c, on the arcs. As seen before, if G is undirected, then we replace each edge by two opposite arcs with the same capacity and cost. To formulate the different problems, for each arc $(i, j) \in A$ we introduce the 0-1 variable y_{ij} and the integer variable x_{ij}, where y_{ij} equals 1 if and only if the arc (i, j) is selected in the solution, and x_{ij} represents the number of terminals connected to the root through the arc (i, j), or equivalently the number of terminals in the subarborescence rooted at j.

We introduce the following polyhedron $\mathcal{T}$:

$$\mathcal{T} = \left\{ x \in \mathbb{N}^{|A|},\ y \in \{0,1\}^{|A|} \;\middle|\; \begin{array}{lll} \sum\limits_{(i,j)\in A} x_{ij} - \sum\limits_{(j,k)\in A} x_{jk} = \begin{cases} |T| & \text{if } j = r \\ -1 & \text{if } j \in T \\ 0 & \text{otherwise} \end{cases} & & \forall j \in V \\ \sum\limits_{(i,j)\in A} y_{ij} \leq 1 & & \forall j \in V \setminus \{r\} \\ x_{ij} \leq u_{ij} y_{ij} & & \forall (i,j) \in A \end{array} \right\}$$

In the following, we write $(x, y) \in \mathcal{T}$ when we consider a couple of variables verifying the constraints of $\mathcal{T}$. The first set of constraints in $\mathcal{T}$ ensures both the conservation of the number of terminals connected through each Steiner vertex $j \in V$ (flow conservation) and the connection of the root to all terminals. The second set of constraints ensures that the solution is an arborescence, i.e., that each vertex has at most one predecessor. Finally, the third set ensures that there is no flow on a non existing arc, and that the number of terminals connected through an arc $(i, j) \in A$ does not exceed its capacity. In the following, the relative gap between two costs will be denoted by Δ. The well-known problem of the Capacitated Steiner Arborescence (**CStA**) can be formulated as follows [20]:

$$\textbf{CStA} \quad \Bigg| \quad \min_{(x,y)\in\mathcal{T}} \sum_{(i,j)\in A} c_{ij} y_{ij}$$

As explained previously, we evaluate the robustness of a Steiner tree by considering the number of terminals disconnected from the root in the worst scenario, that is, the maximum number of terminals connected through an arc incident to the root, which is equal to $\max_{j\in\Gamma^+_G(r)} x_{rj}$. Let R be a fixed bound on this value: we may disconnect at most R terminals from the root by deleting an arc. We propose the following formulation for the Capacitated Steiner Arborescence with bounded robustness ($\textbf{CStA}_{bounded-robust}$):

$$\textbf{CStA}_{bounded-robust} \quad \Bigg| \quad \begin{array}{lll} \min\limits_{(x,y)\in\mathcal{T}} & \sum\limits_{(i,j)\in A} c_{ij} y_{ij} & \\ \text{s.t.} & x_{rj} \leq R & \forall j \in \Gamma^+(r) \end{array}$$

Let us now consider the robustness as an objective. Note that the default objective function

is to minimize the cost of the solution. If a model uses another objective function, then its name will start by a given letter, e.g., R if we want to optimize the worst-case robustness. We propose the following formulation for **RCStA**:

$$\textbf{RCStA} \quad \Bigg| \quad \min_{(x,y)\in\mathcal{T}} \quad \max_{j\in\Gamma_G^+(r)} \; x_{rj}$$

The max function is handled in our formulation with the addition of a variable η with $\eta \geq x_{rj} \quad \forall j \in \Gamma_G^+(r)$: the objective function is then to minimize η. Since this formulation does not take the cost into account, we also propose a new formulation where we bound the cost of a solution by a given value C:

$$\textbf{RCStA}_{bounded-cost} \quad \Bigg| \quad \begin{array}{lll} \min\limits_{(x,y)\in\mathcal{T}} & \max\limits_{j\in\Gamma_G^+(r)} & x_{rj} \\ \text{s.t.} & \sum\limits_{(i,j)\in A} c_{ij}y_{ij} & \leq \; C \end{array}$$

The max function is handled in $\textbf{RCStA}_{bounded-cost}$ in the same way than in **RCStA**.

However, the previous models only consider the worst-case of a breakdown. It appears that it could also be interesting to "balance" the tree in order to reduce the loss due to an "average breakdown". To this end, we consider arc failures at each vertex and not only at the root, i.e., for each $i \in V$, we consider the worst case of a breakdown of an arc leaving i. This corresponds, for each $i \in V$, to the maximum number of terminals that cannot be reached from the root in case of a breakdown of an arc (i, j), $j \in \Gamma_G^+(i)$, or equivalently to the maximum flow on an arc (i, j), $j \in \Gamma_G^+(i)$. We define the "balanced robustness" as the sum of these values: $\sum_{i\in V} \max_{j\in\Gamma_G^+(i)} \; x_{ij}$. This function appears to be an alternative formulation of the losses in both the worst and the average breakdown robustness.

We will use the letters BR to refer to models where one wants to optimize the balanced robustness. We propose formulations similar to the previous ones for the Capacitated Steiner Arborescence with bounded balanced robustness, where we bound the balanced robustness of a solution by a given value BR:

$$\mathbf{CStA}_{bounded-balanced\_robust} \quad \left| \begin{array}{ll} \min\limits_{(x,y)\in\mathcal{T}} & \sum\limits_{(i,j)\in A} c_{ij}y_{ij} \\ \text{s.t.} & \sum\limits_{i\in V} \max\limits_{j\in\Gamma_G^+(i)} x_{ij} \leq BR \end{array} \right.$$

The max function in the constraint is handled in $\mathbf{CStA}_{bounded-balanced\_robust}$ with the addition of $|V|$ variables β_i for each vertex i of V with $\beta_i \geq x_{ij} \quad \forall j \in \Gamma_G^+(i)$: the sum of the variables β_i with $i \in V$ must then not exceed BR.

The following formulation aims at computing the best balanced robustness:

$$\mathbf{BRCStA} \quad \left| \min_{(x,y)\in\mathcal{T}} \sum_{i\in V} \max_{j\in\Gamma_G^+(i)} x_{ij} \right.$$

The max function is handled in **BRCStA** in a similar way than in $\mathbf{CStA}_{bounded-balanced\_robust}$: we minimize the sum of the variables β_i instead of setting an upper bound to this sum.

Moreover, we can keep this latter objective while bounding both the worst-case robustness (by R) and the cost of the solution (by C). We obtain:

$$\mathbf{BRCStA}_{bounded-robust-cost} \quad \left| \begin{array}{lll} \min\limits_{(x,y)\in\mathcal{T}} & \sum\limits_{i\in V} \max\limits_{j\in\Gamma_G^+(i)} x_{ij} & \\ \text{s.t.} & x_{rj} \leq R & \forall j \in \Gamma_G^+(i) \\ & \sum\limits_{(i,j)\in A} c_{ij}y_{ij} \leq C & \end{array} \right.$$

The max function in the objective function of $\mathbf{BRCStA}_{bounded-robust-cost}$ is handled in the same way than in **BRCStA**.

We tested those formulations on real wind farm data sets. Even if the number of instances is small, the results are interesting to analyze, and we can compare the robustness, costs and structures of the solutions. Data parameters and results are available respectively in Tables 2.1a and 2.1b. Figures 2.2, 2.3, 2.4 and 2.5 allow to visually compare the arborescences obtained according to the different models for the fourth data set (the filled circles correspond to terminals).

Figure 2.2 gives an optimal (non robust) capacitated Steiner arborescence (optimal solution of **CStA**); let us denote its cost by C^*. This arborescence cannot be qualified as robust since, in the worst case, all terminals can be disconnected by deleting the only arc incident to the root. Furthermore, the tree has a large depth, and hence the balanced robustness is not good either. This proves the importance of searching for a more robust solution. We consider first the worst case, **RCStA**, and we denote by R^* the best robustness, i.e., the minimum value of the loss of terminals in the worst case of a single arc deletion. See Figure 2.3 for the associated solution on the test instance. Then, to obtain the minimum cost of a most robust solution, denoted by $C^*_{R^*}$, we solve $\mathbf{CStA}_{bounded-robust}$ with $R = R^*$: notice that the constraint is saturated in any feasible solution. Then, $\Delta_{Crob} = (C^*_{R^*} - C^*)/C^*$ represents the "cost of robustness", i.e., the percentage of augmentation of the cost to get a robust solution.

In the same way, let BR^* be the best balanced robustness (optimal value of **BRCStA**, not given in the table); see Figure 2.4 for the associated solution on the test instance. The cost of a solution with the best balanced robustness, denoted by $C^*_{BR^*}$, is obtained by solving $\mathbf{CStA}_{bounded-balanced\_robust}$ with $BR = BR^*$, and $\Delta_{Cbrob} = (C^*_{BR^*} - C^*)/C^*$ represents the "cost of balanced robustness", i.e., the percentage of augmentation of the cost of a non robust arborescence to get a balanced robust solution.

We also study the behavior of the robustness when we bound the cost to a value close to the one of an optimal non robust arborescence : R_8 (resp. R_{12}) corresponds to the optimal value of $\mathbf{RCStA}_{bounded-cost}$ with a bound $C = 1.08C^*$ (resp. $C = 1.12C^*$).

We now analyze the results. The cost of robustness is quite variable on those instances (from 9 to 24%) but remains rather low. On the contrary, we can see that the optimization of the average robustness is way more expensive (raise from 33% to 64% of the cost) because it involves significantly more edges (see Figure 2.4).

As we can see on Table 2.1b, a cost augmentation of 8% or 12% on the optimal cost can result in a solution with a good value of worst-case robustness for some instances: instances 2 and 4 present an excellent value of such robustness with only a cost augmentation of 8%, while instances 1 and 3 have a rather good one with a cost augmentation of 12%.

Finally, we compare the optimal robustness R^* to the robustness of the balanced arborescence S_b obtained by solving **BRCStA**, i.e., we compute in S_b (see Figure 2.4) the maximum

number of terminals which are disconnected after the deletion of an arc incident to the root. Let R_{BR^*} be this number, shown in the last column of Table 2.1b. For the test instances, the values of R^* and R_{BR^*} are the same, which means that S_b is a good solution for both the worst and balanced robustness, but we have seen before that its cost is high. Indeed, for these instances, we see that forcing a solution with $R = R^*$ to be optimally balanced increases the cost by at least 33 %. Nevertheless, there is no guarantee in the general case that the best balanced solution also has the best robustness in the worst case, although the arcs incident to the root are involved in the computation of the balanced robustness.

Table 2.1 Results on robust arborescences and data parameters

Set	|V|	|E|	|T|
1	91	220	42
2	143	382	40
3	220	510	88
4	255	662	73

(a) Data parameters

Set	R^*	R_8	R_{12}	Δ_{Crob}	Δ_{Cbrob}	R_{BR^*}
1	21	35	29	0.18	0.56	21
2	20	21	20	0.09	0.64	20
3	22	32	30	0.24	0.33	22
4	37	41	38	0.19	0.37	37

(b) Results on robust arborescences

When trying to minimize the number of disconnected terminals in the worst case (see **RC-StA** in Figure 2.3), we have seen that the associated solutions have a reasonable cost, but the average robustness is not good, since the tree remains too deep. When finding the Balanced Steiner arborescence (see **BRCStA** in Figure 2.4), the balanced robustness is optimal and the robustness in the worst case is fine, but the cost can be really high (a raise of the optimal cost to 64% on those data sets). Adding bounds on both cost and worst-case robustness, while minimizing the balanced robustness (see $\mathbf{BRCStA}_{bounded-robust-cost}$ in Figure 2.5), yields a solution which has both a reasonable cost and a really good worst-case and balanced robustness, and hence it seems that it actually yields the best compromise between the three optimization criteria (the cost and the two types of robustness considered here).

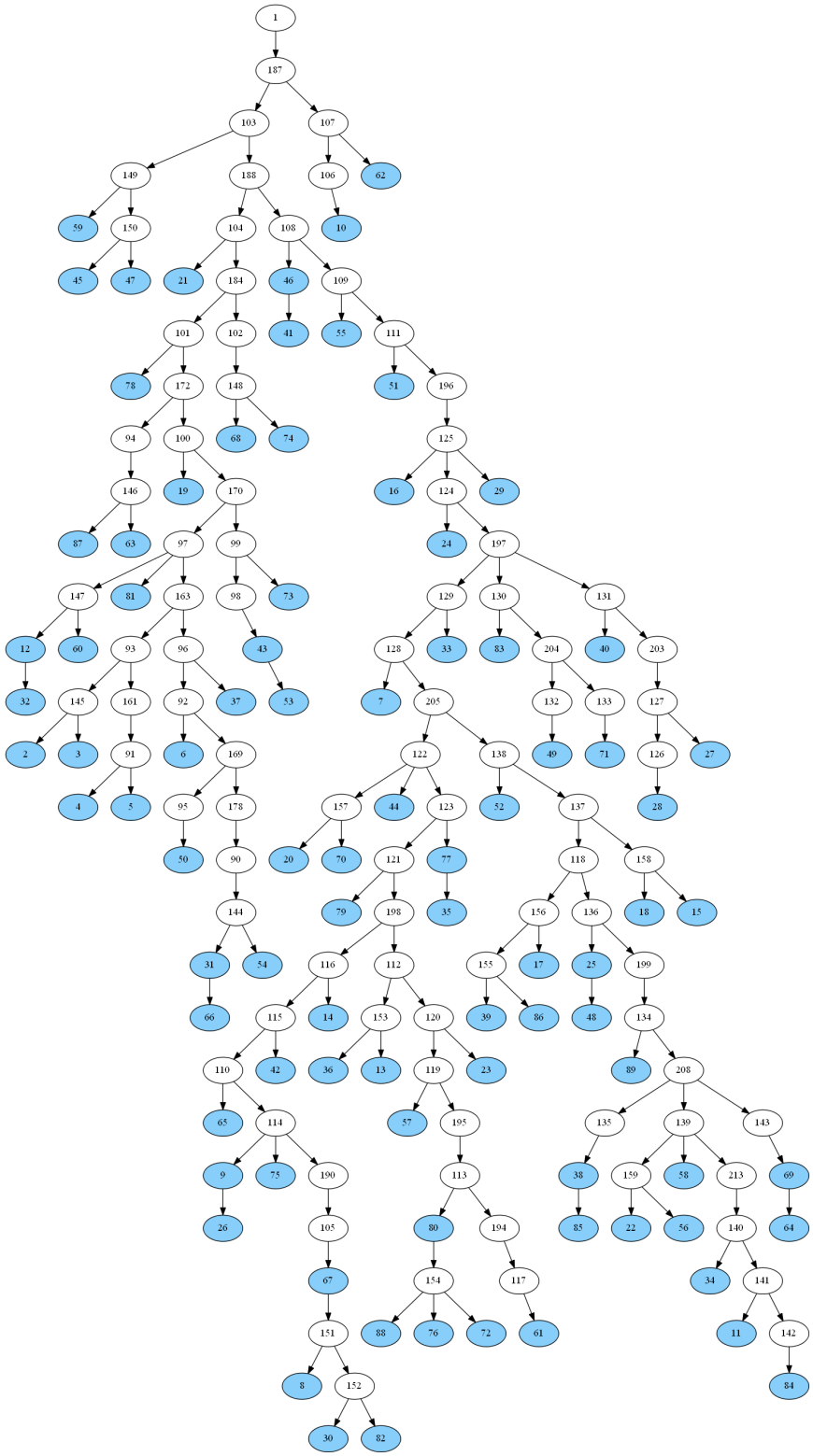


Figure 2.2 Resulting arborescence for the fourth data set for $CStA$

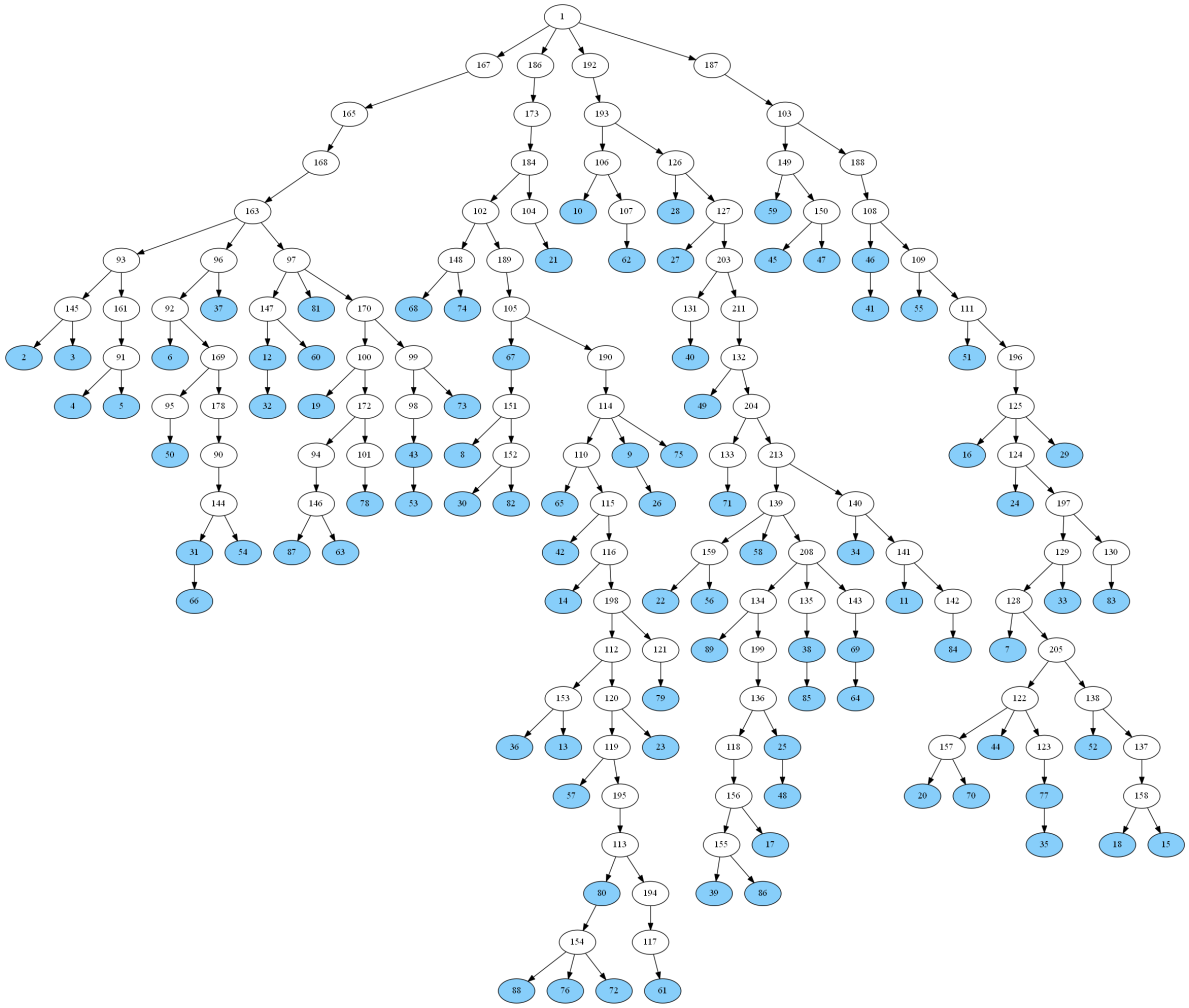


Figure 2.3 Resulting arborescence for the fourth data set for $RCStA$

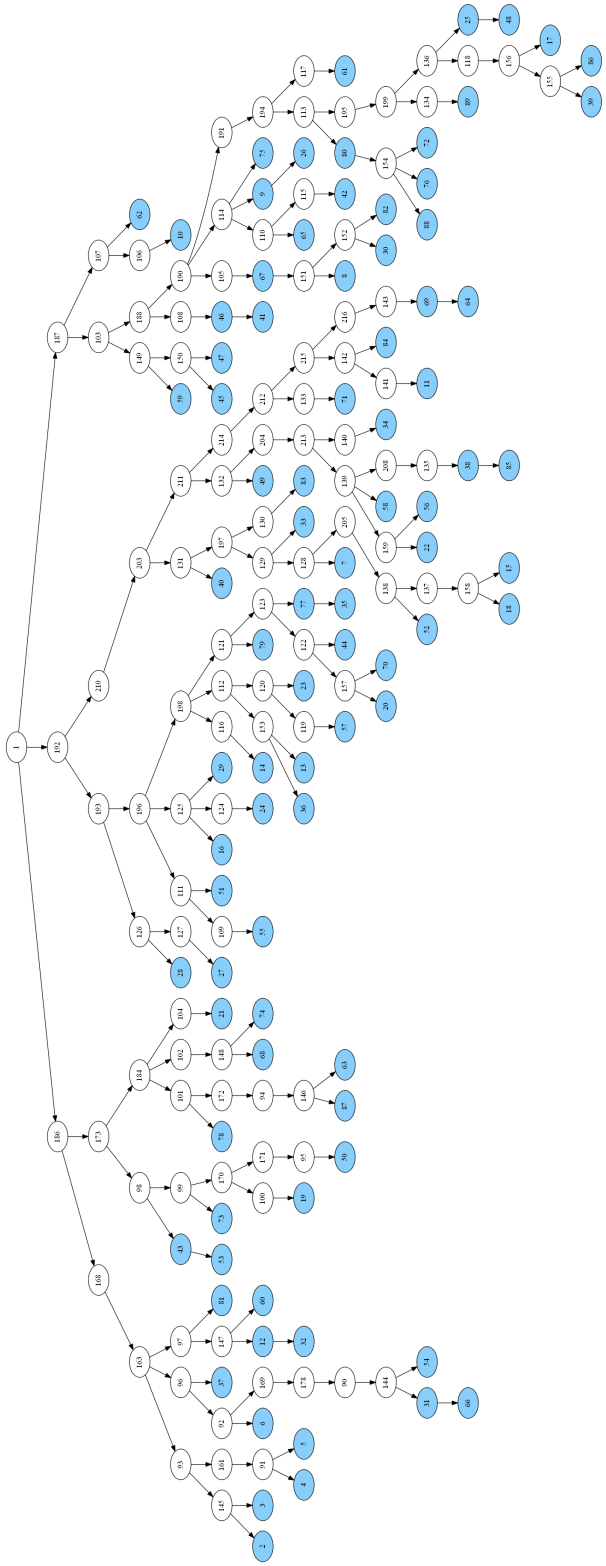


Figure 2.4 Resulting arborescence for the fourth data set for $BRCStA$

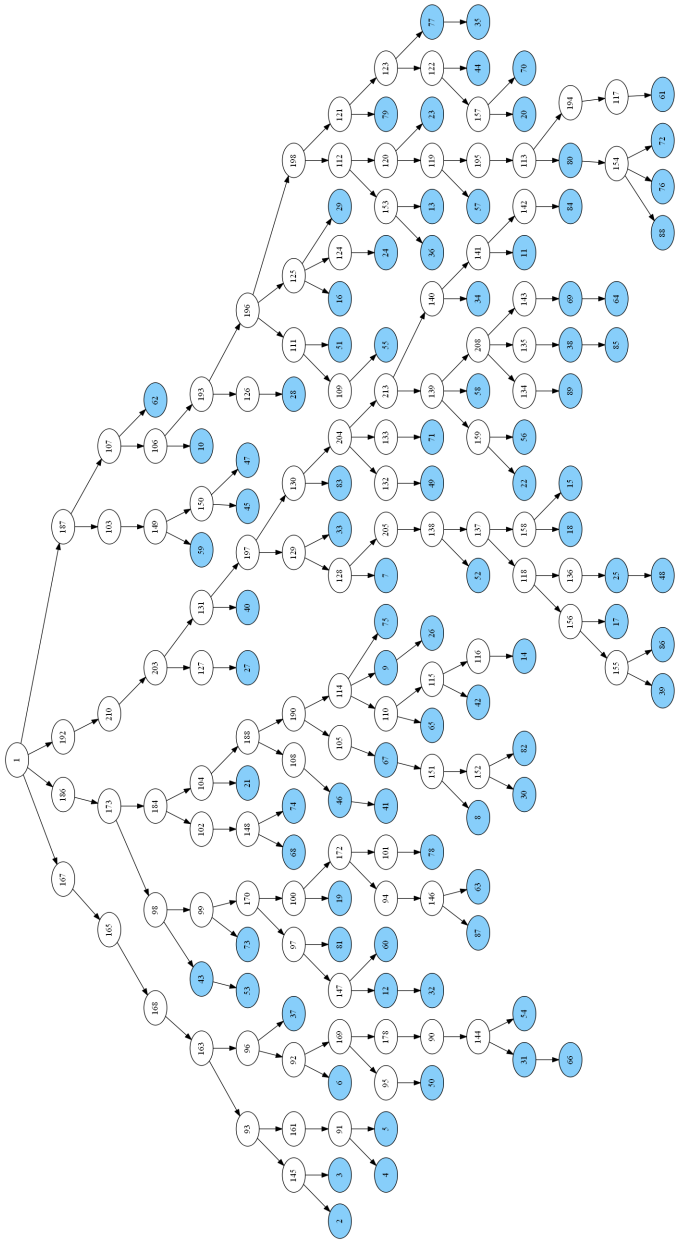


Figure 2.5 Resulting arborescence for the fourth data set for $BRCStA_{bounded-robust-cost}$

CHAPTER 3 CAPACITATED ROOTED k-EDGE CONNECTED STEINER NETWORK PROBLEM (CRkECSN)

3.1 Definitions and notations

In this section, we study the problem of designing networks which are resilient to a given number of arc-failures. A feasible solution to the problem we consider is a network rooted at a given root and covering a given set of terminals, and such that, after deleting any set of k arcs, it is still possible to route a unit of flow from the root to each terminal (see Subsection 1.2.3 in the introduction), while respecting given capacities on the arcs. Formally, we define the following problem:

Capacitated Rooted k-Edge Connected Steiner Network problem (CRkECSN)

INSTANCE: A connected digraph $G = (V, A, r, T, u, c)$ with a set of vertices V, a set of arcs A, a root $r \in V$, a set of terminals $T \subseteq V \setminus \{r\}$, an integer capacity function u on A, a cost function c on A, an integer k with $1 \leq k \leq |A| - 1$.

QUESTION: Find a subset $A' \subseteq A$ of minimum cost such that there is a feasible flow (i.e. respecting the arc capacities) routing a unit of flow from r to each vertex of T in the subgraph of G induced by A', even if any set of k arcs in A' is deleted.

Since we consider that we route one unit of flow from the root to each terminal, it is equivalent to consider that the capacity u_{ij} corresponds to the maximum number of terminals that can be linked to the root through the arc (i, j).

Property 3.1.1 *For $k \in \mathbb{N}^*$, there are at least $k + 1$ arc-disjoint paths between the root and each terminal in any feasible solution. Furthermore, any inclusion-wise minimal feasible solution induces at least a 2-edge-connected graph in the underlying undirected graph.*

Proof: The first part of the property is a consequence of Theorem 1.2.1 ([48]), in Subsection 1.2.1. Let G' be an inclusion-wise minimal feasible solution and assume that G' is not 2-edge-connected in the underlying undirected graph. Then there exists at least one edge e whose removal cuts G' into two parts. If the part that does not include the root contains terminals, then G' is clearly not a feasible solution because, if we remove e, then at least one terminal

cannot be reached from the root. Otherwise, G' is not inclusion-wise minimal because, if we remove e, then the resulting graph is still a feasible solution. Hence, any inclusion-wise minimal feasible solution induces at least a 2-edge-connected graph. □

Remark 3.1.1 *Property 3.1.1 implies that a necessary condition for the existence of a feasible solution is that there are at least $k+1$ arc-disjoint paths between the root and each terminal in G. We assume without loss of generality that this is always verified in G, otherwise there is no feasible solution.*

In order to simplify the formulations proposed in the next sections, we add to the input graph a vertex s (which corresponds to a fictive sink) connected to every terminal $t \in T$ by a fictive arc (t, s) with $c_{ts} = 0$ and $u_{ts} = 1$. Then, s is added to V and the fictive arcs are added to A, and we denote by A_I the set of initial arcs (see Figure 3.1 for an example). We have the following fact:

Fact 3.1.1 *Finding a flow which routes one unit of flow between r and each terminal in the input graph is equivalent to finding a flow of value $|T|$ from r to s in the transformed graph with the sink and the fictive arcs.*

For any partition of $V \setminus \{s\}$ into two parts S_1 and $V \setminus (\{s\} \cup S_1)$ with $r \notin S_1$ and $S_1 \cap T \neq \emptyset$, we must have that $\sum_{(i,j)\in\delta^-(S_1)} u_{ij} \geq |S_1 \cap T|$ in order to allow the routing of a unit of flow from the root to each terminal (we remind that $\delta^-(S_1)$ is the set of arcs entering S_1, see Subsection 1.2.1). In the transformed graph, for any partition of V into two parts S_2 and $V \setminus S_2$ with $r \notin S_2$ and $s \in S_2$, we must have that $\sum_{(i,j)\in\delta^-(S_2)} u_{ij} \geq |T|$ (any cut must have a capacity at least equal to $|T|$ or equivalently there exists a $r - s$ flow of value $|T|$) in order to be able to route a unit of flow from the root to each terminal. In the example proposed in Figure 3.1, we have $T = \{t_1, t_2, t_3\}$ and we call u_i the capacity of the arc e_i for $i = 1, .., 8$. If we take $S_1 = \{r, v_1, t_1\}$, in the input graph we must have that $u_3 + u_4 \geq 2$ while in the transformed graph we must have $u_3 + u_4 + u_8 \geq 3$. Since e_8 is a fictive arc, we have $u_8 = 1$, thus the constraints are equivalent.

3.2 Formulations

3.2.1 Cutset formulation

We introduce, for each $(i, j) \in A$, a binary variable y_{ij} equal to 1 if the arc (i, j) is selected in A', 0 otherwise. In this chapter, the variable y is defined for different formulations and for

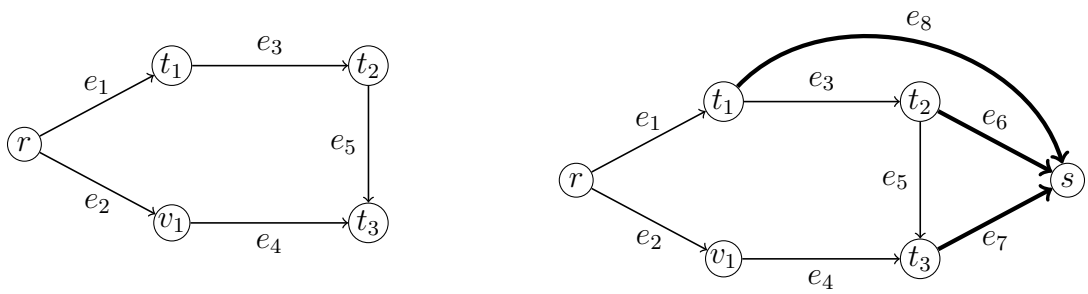


Figure 3.1 Example of addition of a sink to the input graph

each one, y_{ts} is set to 1 for each terminal $t \in T$: the fictive arcs will always be selected in the final network (furthermore their cost is equal to 0 and they cannot fail).

Consider the $r - s$ cuts $[V \setminus V_S, V_S]$ with $V_S \subset V$, $r \in V \setminus V_S$, $s \in V_S$ and $V_S \neq \{s\}$, and let $\mathcal{S}$ be the set of all the associated cut-sets S in A, i.e. $S = \delta^-(V_S)$ for each V_S. $\mathcal{S}$ is the set of $r - s$ cutsets except the one containing only fictive arcs. Notice that if $S \in \mathcal{S}$ then $S \cap A'$ is a cut-set in the selected network. For any set $S \in \mathcal{S}$, let C_k^S be the set of subsets of S of size k of non-fictive arcs. Please note that there are always at least $k + 1$ non-fictive arcs in S because, from Remark 3.1.1, C_k^S cannot be empty. For each $S \in \mathcal{S}$, we define M_S as the maximum capacity of a subset of k selected arcs of S:

$$M_S \quad = \quad \max_{C \in C_k^S} \sum_{(i,j) \in C} u_{ij} y_{ij} \tag{3.1}$$

M_S corresponds to the maximum capacity that can be lost in the cut-set S after the deletion of k arcs. We propose the following cutset formulation:

$$(CUT) \left| \begin{array}{lll} \min\limits_{y} & \sum\limits_{(i,j) \in A} c_{ij} y_{ij} & \\ \text{s.t.} \sum\limits_{(i,j) \in S} u_{ij} y_{ij} \; - \; M_S \; \geq \; |T| & & \forall S \in \mathcal{S} \\ \qquad y_{ij} \in \{0, 1\} & & \forall (i,j) \in A \end{array} \right. \tag{3.2}$$

Constraints (3.2) ensure that, for each cut, the capacity of the cut after the worst-case deletion of k arcs of the cut-set is at least equal to the number of terminals, i.e., one can still route $|T|$ units of flow from r to s while respecting the capacity constraints. They are necessary to every feasible solution. Indeed, they ensure that, for each cut-set in the graph induced by the arcs (i, j) such that $y_{ij} = 1$, the capacity of the cutset minus the k maximal arc capacities

of the cut is greater than or equal to the number of terminals. If a constraint (3.2) is not satisfied for some S, it means that, the capacity of a cut (and consequently of a min-cut) in the graph induced by y after removing k arcs becomes smaller than $|T|$. Constraints (3.2) are also sufficient to ensure a feasible solution: if they are satisfied for each cut-set $S \in \mathcal{S}$, it means that you cannot find a set of k arcs whose removal will induce a min-cut with capacity smaller than $|T|$ (which from Theorem 1.2.2 [29] in Subsection 1.2.1 is a necessary and sufficient condition for the existence of a flow of value $|T|$).

Constraints (3.2) are non linear because of the use of the maximum operator in the definition of M_S. To linearize it, we can rewrite (3.2) as follows:

$$\sum_{(i,j)\in S\setminus C} u_{ij}y_{ij} \quad \geq \quad |T| \qquad \forall S \in \mathcal{S},\ \forall C \in C_k^S \tag{3.3}$$

The number of Constraints (3.3) is obviously bigger than the number of Constraints (3.2). The number of Constraints (3.3) being exponential, we propose a constraints generation algorithm. We begin with a small number of Constraints (3.3), associated with a small subset of $\mathcal{S}$. We obtain a lower bound for our problem. Then we search for a cut-set that does not verify some constraint (3.3) by solving the following subproblem: given a network induced by the arcs (i, j) such that $\hat{y}_{ij} = 1$ (where $\hat{y}$ is the current value of y), we aim to find a cut-set S of minimum residual capacity once we delete its k most capacitated arcs. If this capacity is smaller than $|T|$, we add the constraints associated with S, otherwise the algorithm terminates. For small values of k, one straightforward method to find this cut of minimum residual capacity is the following: for each combination C of k arcs in A_I which are selected in the current solution, compute the min-cut on the graph where the capacity of each arc (i, j) is defined as $u_{ij}\hat{y}_{ij}$, except for the k arcs of C whose capacities are set to 0. Otherwise, if k is too big, we use the following MIP, where $\hat{A} = \{(i, j) \in A \mid \hat{y}_{ij} = 1\}$ and $\hat{G} = (V, \hat{A})$ corresponds to the current solution:

$$
(CUT-SEP)\left|\begin{array}{llll}
\min\limits_{s,b,v} & \sum\limits_{(i,j)\in\hat{A}} u_{ij}s_{ij} & & \\
\text{s.t} & s_{ij}+b_{ij}-v_i+v_j & \geq\ 0 & \forall(i,j)\in\hat{A} \quad (3.4a)\\
 & v_r & =\ 1 & (3.4b)\\
 & v_s & =\ 0 & (3.4c)\\
 & \sum\limits_{(i,j)\in\hat{A}} b_{ij} & \leq\ k & (3.4d)\\
 & \sum\limits_{t\in T} b_{ts} & =\ 0 & (3.4e)\\
 & s_{ij},b_{ij}\ \in\ \{0,1\}\quad \forall(i,j)\in\hat{A}, & v\ \in\ \{0,1\}^{|V|} & (3.4f)
\end{array}\right.
$$

In this MIP, the variable v defines a $r-s$-cut on the network: any vertex j with $v_j = 1$ is in the same part of the cut as r, and any vertex i with $v_i = 0$ is in the same part as s. The variables b_{ij} and s_{ij} are binary and we have $b_{ij} = 1$ if and only if arc (i, j) is deleted, whereas s_{ij} defines the selection of (i, j) in the cut-set: $s_{ij} = 1$ if and only if the arc (i, j) is selected in the cut-set. Constraints (3.4a) ensure that $\tilde{S} = \{(i, j) \text{ s.t. } s_{ij} = 1 \text{ or } b_{ij} = 1\}$ defines a cutset in the current network $\hat{G}$ (if $b_{ij} = 1$, we have $s_{ij} = 0$ in any optimal solution because of the minimization of the objective function), and the objective function $\sum_{(i,j)\in\hat{A}} u_{ij}s_{ij}$ represents the residual capacity of $\tilde{S}$, i.e., the capacity of the undeleted arcs of $\tilde{S}$. Constraints (3.4b) and (3.4c) ensure that the root and the sink are not in the same part of the cut. Constraints (3.4d) and (3.4e) ensure that there are no more than k arc deletions, and that no fictive arc can be deleted. If the solution provides a cutset with a residual capacity at least equal to $|T|$, then the solution is feasible. Otherwise, we add the associated constraint to the main MIP.

In the case of a uniform capacity U on each arc $a \in A_I$, M_S in Constraints (3.2) becomes a constant equal to kU and hence these constraints are linear, which implies that Constraints (3.3) are useless. The number of constraints is still exponential, but highly reduced compared to the non-uniform case. The formulation can be rewritten as follows:

$$
\begin{array}{llll}
\min\limits_{y} & \sum\limits_{(i,j)\in A} c_{ij}y_{ij} & & \\
\text{s.t.} & \sum\limits_{(i,j)\in S} u_{ij}y_{ij} \geq\ |T|\ +\ kU & \forall S\in\mathcal{S} & (3.5)\\
 & y_{ij}\in\{0,1\} & \forall(i,j)\in A &
\end{array}
$$

Undirected graphs

Adapting the formulation to the undirected case where we are given a set of edges E instead of a set of arcs A is quite straightforward. Indeed, it can be done by considering the undirected cut-sets of the graph instead of the directed ones.

3.2.2 Flow formulation

In this section, we introduce a formulation based on flow variables. We define $\mathcal{F}$ as the set of all possible arc-failure scenarios: it corresponds to the set of all k-combinations in A_I. We introduce the variable x_{ij}^F which represents the amount of flow routed through the arc $(i,j) \in A$ when the scenario $F \in \mathcal{F}$ occurs (we have $x_{ij}^F = 0$ if $(i,j) \in F$). The variable y is defined as in the previous formulation (see Subsection 3.2.1). We propose the following flow formulation:

$$
(FLOW) \left|
\begin{array}{llll}
\min\limits_{x,y} \sum\limits_{(i,j)\in A} c_{ij} y_{ij} & & & \\
\text{s.t.} \sum\limits_{i\in\Gamma^-(j)} x_{ij}^F - \sum\limits_{k\in\Gamma^+(j)} x_{jk}^F = 0 & & \forall j \in V \setminus \{r,s\},\ \forall F \in \mathcal{F} & (3.6a) \\
\sum\limits_{t\in\Gamma^-(s)} x_{ts}^F = |T| & & \forall F \in \mathcal{F} & (3.6b) \\
x_{ij}^F \leq u_{ij} y_{ij} & & \forall (i,j) \in A,\ \forall F \in \mathcal{F} & (3.6c) \\
x_{ij}^F = 0 & & \forall F \in \mathcal{F},\ \forall (i,j) \in F & (3.6d) \\
x \in \mathbb{R}_+^{|A|\times|\mathcal{F}|}, \quad y \in \{0,1\}^{|A|} & & &
\end{array}
\right.
$$

Constraints (3.6a) and (3.6b) ensure that there is a flow of value $|T|$ for each arc-failure scenario $F \in \mathcal{F}$, meaning that we can still route a unit of flow to each terminal after any k arc failures. Constraints (3.6c) ensure that the arc capacities are respected for each arc-failure scenario $F \in \mathcal{F}$. Constraints (3.6d) ensure that, in each scenario $F \in \mathcal{F}$, no flow is routed through deleted arcs. One can notice that the variable x must be an integer (because it corresponds to a number of terminals). However, we relax this integrality constraint. Indeed, for any value of $y \in \{0,1\}^{|E|}$, setting the value of x corresponds to routing a set of flows of value $|T|$ on $|\mathcal{F}|$ different networks with integer capacities. Then, for any given value of $y \in \{0,1\}^{|E|}$, there exists a solution where x is integer (see Theorem 1.2.3 in the Subsection 1.2.2 of the introduction), and hence there exists an optimal solution with x integer.

The number of variables and constraints being exponential for arbitrary values of k, we propose a constraints-and-columns generation algorithm to solve the problem. We begin with a small subset of $\mathcal{F}$. The separation problem is the problem of the k most vital links in a flow network which is $\mathcal{NP}$-hard [56]: we search for the k arcs which, once simultaneously deleted, reduce the most the value of a maximum $s-t$ flow. We use a procedure similar to the one used in Subsection 3.2.1: for small values of k, we compute a maximum $s-t$ flow for each combination of k selected arcs of A_I. If there is a combination of arcs whose deletion results in a maximum $s-t$ flow lower than $|T|$, we add this arc-failure scenario, else the solution is feasible. If k is too big, we use the auxiliary MIP $(CUT-SEP)$, see Subsection 3.2.1.

Undirected graphs

In order to adapt the formulation to the undirected case with a set of edges E instead of a set of arcs A, one can define for each $[i,j] \in E$ the variables y_{ij}, x^F_{ij} and x^F_{ji}. The function Γ^+ and Γ^- are replaced by the function Γ in Constraints (3.6a) and (3.6b), while Constraints (3.6c) and (3.6d) are replaced by:

$$x^F_{ij} + x^F_{ji} \leq u_{ij} y_{ij} \qquad \forall [i,j] \in E,\ \forall F \in \mathcal{F} \tag{3.7a}$$

$$x^F_{ij} + x^F_{ji} = 0 \qquad \forall F \in \mathcal{F},\ \forall [i,j] \in F \tag{3.7b}$$

A feasible solution induced by y and x implies a flow of value $|T|$ for each scenario $F \in \mathcal{F}$. Then, if a given solution yields a strictly positive flow on both x^F_{ij} and x^F_{ji} for a given edge $[i,j]$ and a given scenario F, there exists another flow at least as good as this one in which the flow verifies either $x^F_{ij} = 0$ or $x^F_{ji} = 0$ (see Remark 1.2.1 in Subsection 1.2.2 of the introduction).

3.2.3 Bilevel formulation

The bilevel formulation proposed here is particular in that the second level is a min max problem. It can be seen as a game with a defender and an attacker (corresponding respectively to the leader and the follower).

For each $(i,j) \in A$, we introduce a variable x_{ij} which corresponds to the amount of flow that the defender chooses to route through the arc (i,j). The variable y is defined as in Subsection 3.2.1. We also introduce the binary variables b_{ij}, $\forall (i,j) \in A$: $b_{ij} = 1$ if and only if the attacker chooses to delete the arc (i,j). Moreover, we assume without loss of generality

that there is no arc entering r. Then, we can define the following polyhedron:

$$\mathcal{X}(y,b) = \left\{ \begin{array}{rcll} \sum_{i\in\Gamma^-(j)} x_{ij} - \sum_{k\in\Gamma^+(j)} x_{jk} & = & 0 & \forall j \in V \setminus \{r,s\} \\ x_{ij} & \leq & u_{ij}y_{ij} & \forall (i,j) \in A \\ x_{ij} & \leq & u_{ij}(1-b_{ij}) & \forall (i,j) \in A \\ x_{ij} & \geq & 0 & \forall (i,j) \in A \end{array} \right\} \quad \begin{array}{r} (3.8a) \\ (3.8b) \\ (3.8c) \\ \\ \end{array}$$

This polyhedron $\mathcal{X}(y,b)$ corresponds to the set of possible flows on the subgraph of G induced by the arcs $(i,j) \in A$ such that $y_{ij} = 1$, provided they have not been deleted, i.e. $b_{ij} = 0$. The polyhedron $\mathcal{X}(y,b)$ is defined by the flow conservation constraints, the capacity constraints and the constraints imposing a flow equal to 0 on any arc which is deleted. We also define the following polyhedron:

$$\mathcal{B} = \{\ b \in \{0,1\}^{|A|} \mid \textstyle\sum_{(i,j)\in A} b_{ij} \leq k\ ;\ b_{ts} = 0 \quad \forall t \in T\ \}$$

The polyhedron $\mathcal{B}$ defines the set of possible scenarios of arc failures (it ensures that no fictive arc can be deleted). We propose the following bilevel program:

$$(BILEVEL) \left| \begin{array}{lll} \min\limits_{y\in\{0,1\}^{|A|}} & \sum\limits_{(i,j)\in A} c_{ij}y_{ij} & \\ \text{s.t.} & f(y) \geq |T| & (3.9a) \\ & \text{where } f(y) = \min\limits_{b\in\mathcal{B}} \max\limits_{x\in\mathcal{X}(y,b)} \sum\limits_{j\in\Gamma^+(r)} x_{rj} & (3.9b) \end{array} \right.$$

where $\{(i,j) \text{ s.t. } y_{ij} = 1\}$ defines the set of selected arcs. At the upper level, the defender selects the set of arcs to be added to the network, by choosing a value of y in $\{0,1\}^A$. The attacker then deletes some arcs by setting the variable $b \in \mathcal{B}$ in order to minimize the maximum flow that the defender will compute by setting the variable x in the flow polyhedron $\mathcal{X}(y,b)$. The aim of the defender is to ensure that this flow is at least equal to $|T|$ (see Constraint (3.9a)).

Consider the max problem in the lower level: at this stage, y and b are already fixed; we refer to their values as $\hat{y}$ and $\hat{b}$ respectively. The problem is a max-flow problem from r to s,

with two sets of capacity constraints. In our problem, the flow must be integral since it corresponds to a number of terminals. However, it is well-known that the matrix of coefficients M in the arc-formulation of a max-flow is totally unimodular (see Subsection 1.2.2 in the introduction). Then, adding the second set of capacity constraints is equivalent to appending the identity matrix to M: the matrix remains totally unimodular and, since the capacities are integers, we ensure that the extreme points of the polyhedron defined by $\mathcal{X}(y,b)$ have integral coordinates. Thus, we can relax the integrality constraints on x.

In the max problem of the lower level (i.e. $\max_{x\in\mathcal{X}(y,b)} \sum_{j\in\Gamma^+(r)} x_{rj}$), there always exists a feasible flow of value 0 and the problem is also trivially upper bounded by $|T|$ (because of the cut-set with only the fictive arcs). Hence, the strong duality holds and we can introduce the dual of the lower level problem, where μ is the variable associated with Constraints (3.8a) while λ and γ are respectively associated with Constraints (3.8b) and (3.8c). After a slight reformulation due to the addition of μ_r and μ_s, the dual problem can be written as follows (see Subsection 1.2.2 in the introduction):

$$
(\mathbf{\Phi})\left|\begin{array}{lllll}
\min\limits_{\lambda,\mu,\gamma} & \sum\limits_{(i,j)\in A} u_{ij}\hat{y}_{ij}\lambda_{ij} + \sum\limits_{(i,j)\in A} u_{ij}(1-\hat{b}_{ij})\gamma_{ij} & & & \\
\text{s.t} & \lambda_{ij}+\gamma_{ij}-\mu_i+\mu_j & \geq & 0 & \forall (i,j)\in A \quad (3.10a)\\
 & \mu_r & = & 1 & (3.10b)\\
 & \mu_s & = & 0 & (3.10c)\\
 & \lambda,\gamma \in [0,1]^{|A|}, \qquad \mu \in [0,1]^{|V|} & & & (3.10d)
\end{array}\right.
$$

Because $(\mathbf{\Phi})$ is the dual of the problem $\max_{x\in\mathcal{X}(y,b)} \sum_{j\in\Gamma^+(r)} x_{rj}$, we have that the matrix of constraints of $(\mathbf{\Phi})$ is totally unimodular (because it is the transpose of a totally unimodular matrix, see Property 1.2.1 in Subsection 1.2.2 in the introduction). Thus, the extreme points of $(\mathbf{\Phi})$ are integer. This problem is a special formulation of a min-cut problem: μ defines the two parts of the cut (sets of vertices $i \in V$ such that either $\mu_i = 0$ or $\mu_i = 1$), an arc (i,j) is in the corresponding cut-set if $\mu_i = 1$ and $\mu_j = 0$. The variables γ and λ define the cut-set of the corresponding cut: for each arc (i,j) in the cut-set, we have either $\lambda_{ij} = 1$ or $\gamma_{ij} = 1$. If an arc (i,j) is not in the cut-set, because we minimize the objective function with positive coefficients and $\mu_j - \mu_i \geq 0$, we have $\gamma_{ij} = \lambda_{ij} = 0$ in an optimal solution. Moreover, because of the economic function and the positive capacities, we have that γ_{ij} is equal to 1 for at least all arcs (i,j) in the cut-set with $\hat{b}_{ij} = \hat{y}_{ij} = 1$ (i.e., the arcs selected but deleted), while λ_{ij} is equal to 1 for at least all arcs (i,j) in the cut-set with $\hat{b}_{ij} = \hat{y}_{ij} = 0$ (i.e., the arcs that

are neither selected nor deleted). For other arcs in the cut-set, it does not matter which one is set to 1. We denote by $\mathcal{D}$ the polyhedron defined by dual Constraints (3.10a)−(3.10d).

Since the lower level can be reformulated as a min − min function by using the dual described above, it can then be rewritten as follows:

$$(LL) \left| \begin{array}{ll} \min\limits_{b,\lambda,\mu,\gamma} & \sum\limits_{(i,j)\in A} u_{ij}\hat{y}_{ij}\lambda_{ij} + u_{ij}(1-b_{ij})\gamma_{ij} \\ \text{s.t} & b \in \mathcal{B} \\ & (\lambda,\mu,\gamma) \in \mathcal{D} \end{array} \right.$$

At this point, b is a variable, so the objective function is non-linear. We linearize the terms $b_{ij}\gamma_{ij}$ in a classical way by introducing variables $l_{ij} = b_{ij}\gamma_{ij}$ where l_{ij} verifies the set of constraints defined by $\mathcal{L}(b,\gamma)$:

$$\mathcal{L}(b,\gamma) = \left\{ l \in \mathbb{R}^{|A|} \left| \begin{array}{lcll} l_{ij} & \leq & b_{ij} & \forall (i,j) \in A \\ l_{ij} & \leq & \gamma_{ij} & \forall (i,j) \in A \\ l_{ij} & \geq & \gamma_{ij} - (1-b_{ij}) & \forall (i,j) \in A \\ l_{ij} & \geq & 0 & \forall (i,j) \in A \end{array} \right. \right\}$$

The lower level can then be rewritten as follows:

$$(LL) \left| \begin{array}{ll} \min\limits_{b,\lambda,\mu,\gamma} & \sum\limits_{(i,j)\in A} u_{ij}\hat{y}_{ij}\lambda_{ij} + u_{ij}\gamma_{ij} - u_{ij}l_{ij} \\ \text{s.t} & b \in \mathcal{B} \\ & (\lambda,\mu,\gamma) \in \mathcal{D} \\ & l \in \mathcal{L}(b,\gamma) \end{array} \right.$$

We define the following function g:

$$g(y,\lambda,\gamma,l) = \sum_{(i,j)\in A} u_{ij}y_{ij}\lambda_{ij} + u_{ij}\gamma_{ij} - u_{ij}l_{ij}$$

We can then rewrite the bilevel program as:

$$
\begin{array}{lll}
\min\limits_{y\in\{0,1\}^{|A|}} & \sum\limits_{(i,j)\in A} c_{ij}y_{ij} & \\
\text{s.t} & f(y) \geq |T| & \\
 & \text{where} \quad f(y) = \min\limits_{b,\lambda,\gamma,\mu,l} & g(y,\lambda,\gamma,l) \\
 & \qquad\qquad\qquad\qquad \text{s.t.} & b \in \mathcal{B} \\
 & & (\lambda,\mu,\gamma) \in \mathcal{D} \\
 & & l \in \mathcal{L}(b,\gamma)
\end{array}
$$

We can then consider the convex hull of the lower-level polyhedron defined by $\mathcal{B}$, $\mathcal{D}$ and $\mathcal{L}(b,\gamma)$, and denote by $\mathcal{H}$ the set of its extreme points. One can notice that this convex hull does not depend on y (only $g(\cdot)$ does): the set of extreme points $\mathcal{H}$ remains the same for every $y \in \{0,1\}^A$. We denote by $(\hat{\lambda}^h, \hat{\gamma}^h, \hat{l}^h)$ the respective values of (λ,γ,l) at the extreme point $h \in \mathcal{H}$. We can then reformulate the bilevel formulation as a single-level one as follows:

$$
(\mathbf{BP}) \qquad
\begin{array}{llll}
\min & \sum\limits_{(i,j)\in A} c_{ij}y_{ij} & & \\
\text{s.t.} & g(y,\hat{\lambda}^h,\hat{\gamma}^h,\hat{l}^h) \;\geq\; |T| & \forall h \in \mathcal{H} & (3.11\text{a}) \\
 & y \in \{0,1\}^{|A|} & & (3.11\text{b}) \\
 & b \in \mathcal{B} & & (3.11\text{c}) \\
 & (\lambda,\mu,\gamma) \in \mathcal{D} & & (3.11\text{d}) \\
 & l \in \mathcal{L}(b,\gamma) & & (3.11\text{e})
\end{array}
$$

Constraints (3.11a) ensure that, for each extreme point of $\mathcal{H}$, $f(y)$ is greater than $|T|$ (thus the minimum value of $f(y)$ over the polyhedron defined by Constraints (3.11b)−(3.11e) is greater than $|T|$), meaning that the value of a maximum flow cannot become smaller than $|T|$, even after any k breakdowns.

Remark 3.2.1 *One may think that the sets of constraints $x_{ij} \leq u_{ij}y_{ij}$ and $x_{ij} \leq u_{ij}(1-b_{ij})$ could be replaced by $x_{ij} \leq u_{ij}(y_{ij} - b_{ij})$ in the polyhedron $\mathcal{X}(y,b)$ to make the resulting polyhedron $\tilde{\mathcal{X}}(y,b)$ more compact. However, if for a given (i,j), we have $b_{ij} > y_{ij}$ (i.e. $b_{ij} = 1$ and $y_{ij} = 0$), then $\tilde{\mathcal{X}}(y,b)$ is empty and hence $\min_{b\in\mathcal{B}} \max_{x\in\tilde{\mathcal{X}}(y,b)} \sum_{j\in\Gamma^+(r)} x_{rj}$ is not defined because there are some values of b for which $\tilde{\mathcal{X}}(y,b)$ is empty. Furthermore the dualization of this polyhedron leads (LL) to be unbounded in this case.*
A solution to this issue could be to add the set of constraints $b_{ij} \leq y_{ij}$ in the polyhedron $\mathcal{B}$ but it would lead to the presence of constraints which depend on y in the lower level.

Remark 3.2.2 *In* (**BP**), $g(y, \lambda, \gamma, l)$ *is non-linear because of the products* $y_{ij}\lambda_{ij}$*, but they can be linearized as it has been done for* $b_{ij}\gamma_{ij}$ *above.*

However, there is an exponential number of Constraints (3.11a). To tackle this issue, we use a constraints generation algorithm where we relax Constraints (3.11a) and use (LL) as the separation problem: while the optimum value of (LL) is smaller than $|T|$ for the current solution $\hat{y}$ (integer solution to (**BP**) with a subset of Constraints (3.11a)), we generate Constraints (3.11a) associated with the extreme point whose coordinates are the optimal values of $(b, \lambda, \gamma, \mu, l)$ in (LL).

Property 3.2.1 *Let* $\hat{y}^1$ *and* $\hat{y}^2$ *be two feasible solutions of* (**BP**) *such that* $\hat{y}^1 \geq \hat{y}^2$*, i.e.,* $\hat{y}^1_{ij} \geq \hat{y}^2_{ij}$ *for each arc* (i, j)*. If adding a constraint* $g(y, \hat{\lambda}^a, \hat{\gamma}^a, \hat{l}^a) \geq |T|$ *makes any solution with* $y = \hat{y}^1$ *infeasible, then it also makes any solution with* $y = \hat{y}^2$ *infeasible.*

Proof: For any value $(\hat{\lambda}^a, \hat{\gamma}^a, \hat{l}^a)$ of (λ, γ, l), we have $g(\hat{y}^1, \hat{\lambda}^a, \hat{\gamma}^a, \hat{l}^a) \geq g(\hat{y}^2, \hat{\lambda}^a, \hat{\gamma}^a, \hat{l}^a)$ since $\hat{y}^1 \geq \hat{y}^2$ (recall that u and λ are positive). Hence, if $g(\hat{y}^1, \hat{\lambda}^a, \hat{\gamma}^a, \hat{l}^a) \leq |T| - 1$, then $g(\hat{y}^2, \lambda^a, \gamma^a, l^a) \leq |T| - 1$. □

To improve the cut added to (**BP**) to forbid the current non-feasible solution $\hat{y}$ obtained by solving (LL) for each constraint generation, we search for an unfeasible solution y such that $y \geq \hat{y}$ in order to generate a stronger constraint, as explained in Property 3.2.1. To get these values, we first solve the following problem, and then we compute the new $\hat{y}$ accordingly (as explained later). Given a starting solution $\hat{y}$, we propose to find a cut-set in the support network (i.e., in the initial digraph G) with a minimum number of arcs such that this cut-set is non-valid in the network induced by $\hat{A}$ (i.e. the arcs (i, j) such that $\hat{y}_{ij} = 1$), meaning that, if we remove k given arcs of the cut-set in $\hat{A}$, its remaining capacity is smaller than $|T|$. This can be modeled as follows:

$$
\begin{array}{llll}
\min & \sum_{(i,j)\in A} \lambda_{ij} & & \\
\text{s.t} & \displaystyle\sum_{(i,j)\in A} u_{ij}\hat{y}_{ij}\lambda_{ij} & \leq \; |T| - 1 & \qquad (3.12a) \\
& \displaystyle\sum_{(i,j)\in A} \gamma_{ij} & = \; k & \qquad (3.12b) \\
& \gamma_{ts} & = \; 0 \qquad \forall t \in T & \qquad (3.12c) \\
& (\lambda, \mu, \gamma) \in \mathcal{D}, \; \mu \in \{0,1\}^{|V|} & &
\end{array}
$$

The variables (λ, μ, γ) define a cut in the input graph since they belong to $\mathcal{D}$ (recall that

$\mathcal{D}$ is the set of Constraints (3.10a)−(3.10d)): μ defines the two parts of the cut, while λ and γ define the arcs in the cut-set; in particular γ defines the arcs in the cut-set that are deleted. However, adding the other constraints makes the constraints matrix not unimodular anymore: thus, we have to set μ as a 0-1 variable. When μ is binary, there always exist an optimal solution with γ and λ binary: for an arc (i, j), if $\mu_i = \mu_j$ or $\mu_i = 0$ and $\mu_j = 1$, we have that $\lambda_{ij} = \gamma_{ij} = 0$; if $\mu_i = 1$ and $\mu_j = 0$, we must have $\lambda_{ij} + \gamma_{ij} \geq 1$, and because of the minimization of the objective function, there always exists an optimal solution such that exactly one between λ_{ij} and γ_{ij} is equal to 1. Constraint (3.12a) ensures that the cut-set selected is non-valid (i.e. has insufficient capacity). Constraint (3.12b) ensure that the number of deleted arcs is equal to k, while Constraints (3.12c) forbid the deletion of fictive arcs.

Then, the new values of the $\hat{y}_{ij}$ are computed as follows: we set $\hat{y}_{ij}$ to 1 for all (i, j) with $\lambda_{ij} = \gamma_{ij} = 0$ and let the others to their current value. Indeed, we want to include as many arcs as possible in the solution, while ensuring that there exists a cut with capacity at most |T|-1: hence, any arc not associated with this cut can be included in the solution. It implies that, for each arc (i, j), the new value of $\hat{y}_{ij}$ cannot be smaller than the old one, and, using Proposition 3.2.1, we generate a better constraint than the original one by computing the extreme points associated with this new value of $\hat{y}$.

Undirected graphs

In order to obtain a formulation that works for the undirected case, we define for each edge $[i, j]$ the variables y_{ij}, b_{ij}, x_{ij} and x_{ji}. The only modification appears in the polyhedron $\mathcal{X}(y, b)$, which can be modified as follows:

$$\mathcal{X}(y,b) = \left\{ x \in \mathbb{R}^{|E|} \left| \begin{array}{lcll} \sum\limits_{i \in \Gamma(j)} x_{ij} - \sum\limits_{k \in \Gamma(j)} x_{jk} = 0 & & & \forall j \in V \setminus \{r, s\} \\ x_{ij} & \leq & u_{ij} y_{ij} & \forall [i,j] \in E \\ x_{ji} & \leq & u_{ij} y_{ij} & \forall [i,j] \in E \\ x_{ij} & \leq & u_{ij}(1 - b_{ij}) & \forall [i,j] \in E \\ x_{ji} & \leq & u_{ij}(1 - b_{ij}) & \forall [i,j] \in E \\ x_{ij}, x_{ji} & \geq & 0 & \forall [i,j] \in E \end{array} \right. \right\}$$

Again, since this polyhedron is associated with a maximum flow problem (when the values of y and b are fixed), we can always find a maximum flow where either $x_{ij} = 0$ or $x_{ji} = 0$

for each edge $[i,j] \in E$. Once this polyhedron has been modified, one can use the method proposed for the directed case to solve the formulation.

3.3 Relations between the formulations

3.3.1 Relations between the bilevel and the cutset formulations

Let us formulate (SL) the second level of the bilevel formulation by replacing Constraints (3.8b) and Constraints (3.8c) by the constraint $x_{ij} \leq u_{ij}(\hat{y}_{ij} - b_{ij}) \quad \forall (i,j) \in A$ (see the polyhedron $\tilde{\mathcal{X}}(y,b)$ in Remark 3.2.1 in Subsection 3.2.3). We introduce the polyhedron $\tilde{\mathcal{B}}(y) = \{b \in \mathcal{B} \mid b \leq y\}$ to ensure that $\tilde{\mathcal{X}}(y,b)$ is non-empty for each possible value of $b \in \tilde{\mathcal{B}}(y)$, i.e. the second level is defined. We obtain:

$$
(\mathbf{SL}) \left|
\begin{array}{llll}
\min\limits_{b \in \tilde{\mathcal{B}}(y)} & \max\limits_{x} & \sum\limits_{j \in \Gamma^+(r)} x_{rj} & \\
\text{s.t} & \sum\limits_{i \in \Gamma^-(j)} x_{ij} - \sum\limits_{k \in \Gamma^+(j)} x_{jk} = 0 & \forall j \in V \setminus \{r,s\} & (3.13a) \\
& 0 \leq x_{ij} \leq u_{ij}(\hat{y}_{ij} - b_{ij}) & \forall (i,j) \in A & (3.13b)
\end{array}
\right.
$$

We introduce a reformulation of (SL), moving the variable b to the objective function:

$$
(\mathbf{RSL}) \left|
\begin{array}{llll}
\min\limits_{b \in \tilde{\mathcal{B}}(y)} & \max\limits_{x} & \sum\limits_{j \in \Gamma^+(r)} x_{rj} - \sum\limits_{(i,j) \in A} b_{ij} x_{ij} & \\
\text{s.t} & \sum\limits_{i \in \Gamma^-(j)} x_{ij} - \sum\limits_{k \in \Gamma^+(j)} x_{jk} = 0 & \forall j \in V \setminus \{r,s\} & (3.14a) \\
& 0 \leq x_{ij} \leq u_{ij}\hat{y}_{ij} & \forall (i,j) \in A & (3.14b)
\end{array}
\right.
$$

Property 3.3.1 *(SL) and (RSL) have the same optimal values.*

Proof: Let $f^1(\hat{b},\hat{x})$ (respectively $f^2(\hat{b},\hat{x})$) be the value of the objective function of (SL) (respectively (RSL)) when $(\hat{b},\hat{x})$ is a feasible solution of (SL) (respectively (RSL)).

Let x^1 be an optimal solution of the max problem of (SL) associated with $b' \in \tilde{\mathcal{B}}(y)$. We have that x^1 is obviously a feasible solution of (RSL). Furthermore, Constraints (3.13b) imply that $x^1_{ij} = 0$ if $b'_{ij} = 1$. Then $b'_{ij}x^1_{ij} = 0$ for all arcs (i,j) and $f^1(b',x^1) = f^2(b',x^1)$.

Let x^2 be an optimal solution of the max problem of (RSL) associated with $b'' \in \tilde{\mathcal{B}}(y)$. If $b''_{ij}x^2_{ij} = 0$ for all arcs (i,j), x^2 is a feasible solution of the max problem of (SL) and $f^2(b'', x^2) = f^1(b'', x^2)$. Otherwise there is at least one arc (u,v) such that $b''_{uv}x^2_{uv} > 0$ (thus x^2 is not feasible for the max problem of (SL)). Let x^3 be a new solution of (RSL) such that $x^3_{uv} = 0$, $x^3_{ij} \leq x^2_{ij}$ for each arc $(i,j) \neq (u,v)$ and the value of the flow defined by x^3 is equal to the value of the flow defined by x^2 minus x^2_{uv} (i.e. $\sum_{j\in\Gamma^+(r)} x^3_{rj} = \sum_{j\in\Gamma^+(r)} x^2_{rj} - x^2_{uv}$). Such a flow always exists: we reduce by x^2_{uv} the value of the flow by reducing the amount of flow on one or several paths from r to s containing (u,v).

We have $f^2(b'', x^3) = \sum_{j\in\Gamma^+(r)} x^3_{rj} - \sum_{(i,j)\in A} b''_{ij}x^3_{ij} = (\sum_{j\in\Gamma^+(r)} x^2_{rj} - x^2_{uv}) - \sum_{(i,j)\in A} b_{ij}x^3_{ij}$. Because $x^3 \leq x^2$ and $x^3_{uv} = 0$, we have that $\sum_{(i,j)\in A} b''_{ij}x^2_{ij} \geq \sum_{(i,j)\in A} b''_{ij}x^3_{ij} + x^2_{uv}$ and thus $f^2(b'', x^3) \geq \sum_{j\in\Gamma^+(r)} x^2_{rj} - x^2_{uv} - (\sum_{(i,j)\in A} b_{ij}x^2_{ij} - x^2_{uv})$ which leads to $f^2(b'', x^3) \geq f^2(b'', x^2)$: since x^2 is optimal we have $f^2(b'', x^3) = f^2(b'', x^2)$. We have found an optimal solution x^3 of the max problem of (RSL) with $x^3_{uv} = 0$ and $x^3 \leq x^2$. By using this method iteratively, we obtain a solution x^h with $\sum_{(i,j)\in A} b''_{ij}x^h_{ij} = 0$. Thus, x^h is optimal for (RSL) and feasible for (SL) and $f^2(b'', x^2) = f^2(b'', x^h) = f^1(b'', x^h)$.

The optimal values of (SL) and (RSL) are then the same considering we can transform an optimal solution for (SL) to a solution for (RSL) of same value and vice versa. $\square$

In an optimal solution of (RSL), the defender will then route no flow through arcs which had been deleted by the attacker because it would reduce the value of his objective function. We can also replace the constraint $b \in \tilde{\mathcal{B}}(y)$ by $b \in \mathcal{B}$ in (RSL) (i.e. deleting the constraint $b \leq y$) since the polyhedron of the max problem is defined for each $b \in \mathcal{B}$. This does not affect the optimal value of (RSL) since, for an arc (i,j), $x_{ij} = 0$ if $y_{ij} = 0$ and thus $b_{ij}x_{ij} = 0$ no matter the value of b_{ij}.

Property 3.3.1 can also be seen as a consequence of a Wood's result [67]. Wood considers the following generic bilevel network interdiction model:

$$(BNI1) \left| \begin{array}{lll} \min\limits_{z\in\mathcal{Z}} & \max\limits_{w} & c^\top w \\ \text{s.t} & Aw & \leq a \\ & 0 \leq w & \leq U(1-z) \end{array} \right.$$

where $\mathcal{Z} = \{z \in \{0,1\}^n \mid Hz \ \leq \ h\}$ is the polyhedron of the attacker variable z (for a facility f, $z_f = 1$ if the facility f is attacked, 0 otherwise). The defender operates the network to maximize his objective function by setting the variable w, considering the damages made by the attacker (if $z_f = 1$, there cannot be any activity on the facility f, i.e. $w_f = 0$ because of the constraint $0 \leq w \leq U(1-z)$). We have $U = diag(u)$ where u is the upper bound vector of w (for example a capacity).

Let $\overline{r}_f$ be an upper bound on the optimal dual variable for the constraint $w_f \leq u_f(1-z_k)$ in $(BNI1)$. We introduce $\overline{r} = (\overline{r}_1 ... \overline{r}_n)^\top$ and $\overline{R} = diag(\overline{r})$. Wood defines the following formulation:

$$(BNI2) \left| \begin{array}{lll} \min\limits_{z \in \mathcal{Z}} & \max\limits_{w} & (c^\top - z^\top \overline{R})w \\ \text{s.t} & Aw \ \leq \ a & \\ & 0 \ \leq \ w \ \leq \ u & \end{array} \right.$$

In Wood [67, Proposition 1], Wood shows that $(BNI1)$ and $(BNI2)$ are equivalent in the sense that their optimal values are equal and z^* is an optimal solution for $(BNI2)$ if and only if it is an optimal solution for $(BNI1)$. Using this proposition on (SL), we re-obtain the formulation (RSL) ($\overline{R}$ corresponds to the identity matrix in our case since dual variables can be bounded by 1 similarly to Subsection 3.2.3).

Again, at this point, the variable $\hat{y}$ and $\hat{b}$ are fixed. The defender subproblem (RSL) can then be dualized in this way:

$$\begin{array}{llll} \min\limits_{\lambda,\mu} & \sum\limits_{(i,j)\in A} u_{ij}\hat{y}_{ij}\lambda_{ij} & & \\ \text{s.t} & \lambda_{ij} - \mu_i + \mu_j \quad \geq \quad -\hat{b}_{ij} & \forall (i,j) \in A & (3.15a) \\ & \mu_r \quad = \quad 1 & & (3.15b) \\ & \mu_s \quad = \quad 0 & & (3.15c) \\ & \lambda \ \in \ [0,1]^{|A|}, \qquad \mu \ \in \ [0,1]^{|V|} & & (3.15d) \end{array}$$

The lower level can be reformulated as a $\min - \min$ function by using the dual described

above, thus it can be rewritten as follows:

$$
(\mathbf{DRSL}) \left|
\begin{array}{lll}
\min\limits_{b,\lambda,\mu} & \sum\limits_{(i,j)\in A} u_{ij}\hat{y}_{ij}\lambda_{ij} & \\
\text{s.t} & b \in \mathcal{B} & (3.16a) \\
& \lambda_{ij} + b_{ij} - \mu_i + \mu_j \geq \quad 0 \qquad \forall (i,j) \in A & (3.16b) \\
& \mu_r = \quad 1 & (3.16c) \\
& \mu_s = \quad 0 & (3.16d) \\
& \lambda \ \in \ [0,1]^{|A|}, \qquad \mu \ \in \ [0,1]^{|V|} & (3.16e)
\end{array}
\right.
$$

We can then rewrite the bilevel program as:

$$
\begin{array}{lll}
\min\limits_{y\in\{0,1\}^{|A|}} & \sum\limits_{(i,j)\in A} c_{ij}y_{ij} & \\
\text{s.t} & f(y) \geq |T| & \\
& \text{where} \quad f(y) \quad = \min\limits_{(b,\lambda,\mu)\in\mathcal{D}} \sum\limits_{(i,j)\in A} u_{ij}y_{ij}\lambda_{ij} &
\end{array}
$$

where $\mathcal{D}$ is the polyhedron defined by Constraints (3.16a-3.16e).

One can notice that $(DRSL)$ is strictly equivalent to the separation problem of the cut-set formulation (3.4a-3.4e). Let us rewrite Constraint (3.3) from the cut-set formulation:

$$
\sum_{(i,j)\in S\setminus C} u_{ij}y_{ij} \quad \geq \quad |T| \qquad \forall S \in \mathcal{S},\ \forall C \in C_k^S
$$

where $\mathcal{S}$ is the set of $r-s$ cutsets except the one implying only fictive arcs and C_k^S is the set of subsets of S of non-fictive arcs of size k, for any set $S \in \mathcal{S}$. We can see that using the method generating the inequalities associated with the extreme points of the convex hull of $(DRSL)$ proposed in Subsection 3.2.3 generates the same inequalities as (3.3), since the extreme points of $(DRSL)$ represented by the values $(\hat{b}, \hat{\lambda}, \hat{\mu})$, which are integer because of the total unimodularity of the matrix of constraints of $(DRSL)$, will represent cuts on the graph given by $\hat{\mu}$: a vertex v is in the same part as the root if $\mu_v = 1$ and in the same part as the sink if $\mu_v = 0$. The residual cut-set $S \setminus C$ is represented by the arcs (i,j) such that $\hat{\lambda}_{ij} = 1$, the deleted arcs C in the cut-set are represented by the arcs (i,j) such that $\hat{b}_{ij} = 1$.

It is then clear that the bilevel formulation collapses with the cut-set formulation, as stated in Theorem 3.3.1.

Theorem 3.3.1 *The optimal values of the continuous relaxations of* $(BILEVEL)$ *and* (CUT) *are equal.*

Thus, we will only give the numerical results obtained for the cut-set and the flow formulations.

3.3.2 Relations between the flow and the cutset formulations

Let us introduce the continuous relaxation of the cutset formulation:

$$(RCF)\left|\begin{array}{lll} \min\limits_{y} & \sum\limits_{(i,j)\in A} c_{ij}y_{ij} & \\ \text{s.t} & \sum\limits_{(i,j)\in S\setminus C} u_{ij}y_{ij} \;\geq\; |T| & \forall S\in\mathcal{S}, \forall C\in C_k^S \\ & 0 \;\leq\; y_{ij} \;\leq\; 1 & \forall (i,j)\in A \end{array}\right.$$

We remind that $\mathcal{S}$ is the set of cut-sets associated with the $r-s$ cuts $[V\setminus V_S, V_S]$ with $V_S\subset V$, $r\in V\setminus V_S$, $s\in V_S$ and $V_S\neq\{s\}$ and that for any $S\in\mathcal{S}$, C_k^S is the set of subsets of non-fictive arcs of S of size k.

We also introduce the continuous relaxation of the flow formulation:

$$(RFF)\left|\begin{array}{lll} \min\limits_{x,y} & \sum\limits_{(i,j)\in A} c_{ij}y_{ij} & \\ \text{s.t.} & \sum\limits_{i\in\Gamma^-(j)} x_{ij}^F \;-\; \sum\limits_{k\in\Gamma^+(j)} x_{jk}^F = 0 & \forall j\in V\setminus\{r,s\},\ \forall F\in\mathcal{F} \\ & \sum\limits_{t\in\Gamma^-(s)} x_{ts}^F \;=\; |T| & \forall F\in\mathcal{F} \\ & x_{ij}^F \;\leq\; u_{ij}y_{ij} & \forall (i,j)\in A,\ \forall F\in\mathcal{F} \\ & x_{ij}^F \;=\; 0 & \forall F\in\mathcal{F},\ \forall (i,j)\in F \\ & x\in\mathbb{R}_+^{|A|\times|\mathcal{F}|}, \quad y\in[0,1]^{|A|} & \end{array}\right.$$

Let us call (y^1, x^1) a feasible solution for (RFF). Setting $y = y^1$ for (RCF) would give a solution with the same value. Let us check that such a solution is feasible for (RCF). Since (y^1, x^1) is feasible for (RFF), there exists a flow of value $|T|$ for each scenario of breakdown of k arcs, considering the capacity $u_{ij}y_{ij}^1$ on each arc (i,j), except obviously the arcs which

are attacked in this scenario which have a capacity equal to 0. Let us assume that y^1 is not a feasible solution for (RCF). Then there is at least one constraint of type $\sum_{(i,j)\in S\setminus C} u_{ij}y^1_{ij} \geq |T|$ which is violated for some $S \in \mathcal{S}$ and $C \in C^s_k$. If such a constraint is violated, it means that there is a $r-s$ cut-set for which the capacity is lower than $|T|$ if we delete k arcs (again the capacity of each arc (i,j) here is $u_{ij}y^1_{ij}$). Then, the scenario of breakdown $F \in \mathcal{F}$ in which those k arcs are deleted would not admit a feasible flow of value $|T|$: a contradiction. So y^1 is a feasible solution for (RCF) with the same value than (y^1, x^1) for (RFF).

Now let us call y^2 a feasible solution for (RCF). Setting $y = y^2$ for (RFF) would give a solution with the same value. Assume there does not exist any x^2 such that (y^2, x^2) is a feasible solution for (RCC). It means that there exists at least a scenario of breakdown $F \in \mathcal{F}$ such that there is no feasible flow of value $|T|$ on the residual network (where we delete the arcs in F) with capacities equal to $u_{ij}y^2_{ij}$ for each arc (i,j). As before, this is impossible since this would involve a residual $r-s$ cut-set with a capacity lower than $|T|$. So there always exists at least one x^2 such that (y^2, x^2) is a feasible solution for (RFF).

We can then transform any solution of (RFF) in a solution of same value in (RCF) and vice versa and deduce Theorem 3.3.2.

Theorem 3.3.2 *The optimal values of the continuous relaxations of the flow formulation (RFF) and the cut-set formulation (RCF) are equal.*

3.4 Addition of protected arcs

Let us now define another version of the problem, where we add the possibility of protecting k' arcs. In this version, in addition to A', we also select a subset $A'_p \subset A'$ with $|A'_p| \leq k'$; those arcs are called *protected arcs* and cannot be deleted by the attacker. The corresponding problem is called Capacitated Protected Rooted k-Edge Connected Steiner Network problem (**CPRkECSN**). In the wind farm application, protecting arcs can be seen as doubling a set of cables under a given budget for example, or protecting cables from a difficult environment (like extreme cold).

Remark 3.4.1 *With the addition of protected arcs, Property 3.1.1 in Section 3.1 does not hold anymore: if some arcs are protected, a feasible solution does not necessarily imply that there are $k+1$ arc-disjoint paths between the root and each terminal t. For example, $k+1$*

paths which are pairwise arc-disjoint except for the fact that they share a common arc (u, v) can be sufficient to ensure that the terminal t can be reached from the root even after any k arc-deletions if the arc (u, v) is protected and capacities are sufficient. Hence, Remark 3.1.1 can also be ignored.

For each non-fictive arc $(i, j) \in A_I$, we define the variable p_{ij} as a binary variable equal to 1 if the arc (i, j) is protected, and to 0 otherwise. We also define the following polyhedron:

$$P = \{ \ p \in \{0,1\}^{|A_I|} \mid \sum_{(i,j)\in A_I} p_{ij} \leq k' \ ; \ p_{ij} \ \leq \ y_{ij} \quad \forall (i,j) \in A_I \ \}$$

This set ensures that there are at most k' protected arcs, and that we cannot protect arcs which are not selected in the final network. In the following, we propose small modifications to each one of the previous formulations in order to include the possibility of protecting arcs.

3.4.1 Cut-set formulation

In order to include the possibility of protecting arcs to the cut-set formulation proposed in Subsection 3.2.1, Constraints (3.3) can be replaced by the following ones:

$$\sum_{(i,j)\in S} u_{ij} y_{ij} \ - \ \sum_{(i,j)\in C} u_{ij}(y_{ij} - p_{ij}) \quad \geq \quad |T| \qquad \forall S \in \mathcal{S}, \ \forall C \in C_k^S \tag{3.17}$$

Constraints (3.17) ensure that the capacity of each cut-set minus the capacity of k unprotected arcs of this cut-set is always larger than $|T|$. We also add to the cut-set formulation the constraint $p \in P$.

We solve the resulting MIP using the same constraints generation algorithm as in Subsection 3.2.1. The separation problem is slightly modified to take into account the fact that the capacity of the protected arcs cannot be removed to compute the residual capacity of the cut-set. For small values of k, for each combinations of k selected but non-protected arcs, we compute the min-cut (in Subsection 3.2.1, we take into account all selected arcs). Considering the MIP $CUT - SEP$ (see Subsection 3.2.1) used to solve the subproblem, we have to modify the objective function to find a cut-set that is violating some Constraint 3.17, which results

in:

$$
(CUT-SEP-PROT)\left|\begin{array}{llll}
\min\limits_{s,b,v} & \sum\limits_{(i,j)\in\hat{A}}(u_{ij}s_{ij}+u_{ij}\hat{p}_{ij}b_{ij}) & & \\
\text{s.t} & s_{ij}+b_{ij}-v_i+v_j & \geq\ 0 & \forall(i,j)\in\hat{A} \quad (3.18a)\\
 & v_r & =\ 1 & (3.18b)\\
 & v_s & =\ 0 & (3.18c)\\
 & \sum\limits_{(i,j)\in\hat{A}} b_{ij} & \leq\ k & (3.18d)\\
 & \sum\limits_{t\in T} b_{ts} & =\ 0 & (3.18e)\\
 & s_{ij},b_{ij}\ \in\ \{0,1\}\ \forall(i,j)\in\hat{A},\ v\ \in\ \{0,1\}^{|V|} & & (3.18f)
\end{array}\right.
$$

where $\hat{p}$ corresponds to the current value of p. We remind that v defines the partition of the vertex set while s and b define the arcs of the cut-set, in particular b defines all the arcs of the cut-set which are deleted. The objective value is equal to the sum of the capacity of the arcs of the cut-set which are not deleted plus the capacity of the arcs which are deleted but were protected. There always exists an optimal solution where $b_{ij}\hat{p}_{ij}=0$ for each arc (i,j): if there is some $\hat{p}_{ij}b_{ij}>0$, we can set $s_{ij}=1$ and $b_{ij}=0$ which results in a solution with the same objective value which still satisfies all the constraints. Thus, we can add the constraints

$$b_{ij}\leq 1-\hat{p}_{ij} \qquad \forall(i,j)\in\hat{A} \tag{3.19}$$

and set the objective function to

$$\min_{s,b,v}\sum_{(i,j)\in\hat{A}} u_{ij}s_{ij}$$

Thus, protected arcs cannot be deleted.

Remark 3.4.2 *When arcs can be protected, the case of uniform capacities equal to U does not admit the reformulation (3.5) anymore. For example, for a cut-set $S\in\mathcal{S}$, if all arcs of S are protected, we must just ensure that their capacity is at least equal to $|T|$, whereas if no arc is protected we must ensure that $\sum_{(i,j)\in S}u_{ij}y_{ij}-kU$ is at least equal to $|T|$.*

3.4.2 Equivalence between cut-set and bilevel formulations in the protected case

The bilevel formulation introduced in Subsection 3.2.3 can be adapted to the protected case as follows:

$$(BIL-PROT) \left| \begin{array}{ll} \min\limits_{y\in\{0,1\}^{|A|}} & \sum\limits_{(i,j)\in A} c_{ij}y_{ij} \\ \text{s.t.} & f(y) \geq |T| \\ & \text{where } f(y) = \quad \min\limits_{b\in\mathcal{B}(p)} \quad \max\limits_{x\in\mathcal{X}(y,b)} \quad \sum\limits_{j\in\Gamma^+(r)} x_{rj} \end{array} \right.$$

where $b \in \mathcal{B}(p)$ with

$$\mathcal{B}(p) = \{\ b \in \{0,1\}^{|A|} \mid \sum_{(i,j)\in A} b_{ij} \leq k\ ;\ \sum_{t\in T} b_{ts} = 0\ ;\ b_{ij} \leq 1 - p_{ij} \quad \forall (i,j) \in A\ \}$$

forbids the attacker to delete some protected arcs.

The reformulation of the second level results in:

$$(\mathbf{RSL-Prot}) \left| \begin{array}{lll} \min\limits_{b\in\tilde{\mathcal{B}}(p)} & \max\limits_{x} \quad \sum\limits_{j\in\Gamma^+(r)} x_{rj} \ - \sum\limits_{(i,j)\in A} b_{ij}x_{ij} & \\ \text{s.t} & \sum\limits_{i\in\Gamma^-(j)} x_{ij} - \sum\limits_{k\in\Gamma^+(j)} x_{jk} \ = \ 0 \qquad \forall j \in V \setminus \{r,s\} & (3.21a) \\ & 0 \ \leq \ x_{ij} \ \leq \ u_{ij}\hat{y}_{ij} \qquad \forall (i,j) \in A & (3.21b) \end{array} \right.$$

This leads to the dual reformulated second level problem:

$$
(\mathbf{DRSL-Prot}) \left|
\begin{array}{llll}
\min\limits_{b,\lambda,\mu} & \sum\limits_{(i,j)\in A} u_{ij}\hat{y}_{ij}\lambda_{ij} & & \\
\text{s.t} & \lambda_{ij} + b_{ij} - \mu_i + \mu_j \geq \quad 0 & \forall (i,j) \in A & (3.22a) \\
& \mu_r \quad = \quad 1 & & (3.22b) \\
& \mu_s \quad = \quad 0 & & (3.22c) \\
& \sum\limits_{(i,j)\in A} b_{ij} \quad \leq \quad k & & (3.22d) \\
& \sum\limits_{t\in T} b_{ts} \quad = \quad 0 & & (3.22e) \\
& b_{ij} \quad \leq \quad 1 - \hat{p}_{ij} & \forall (i,j) \in A & (3.22f) \\
& \lambda \in [0,1]^{|A|}, \quad \mu \in [0,1]^{|V|}, \quad b \in \{0,1\}^{|A|} & & (3.22g)
\end{array}
\right.
$$

which is equivalent to $(CUT - SEP - PROT)$ in Subsection 3.4.1. Similarly to Subsection 3.3.1, the two formulations are equivalent as in the unprotected case.

3.4.3 Flow formulation

In the flow formulation, in addition to the constraint $p \in P$, we can replace Constraints (3.6d) by the following ones:

$$
x_{ij}^F \quad \leq \quad u_{ij}p_{ij} \qquad \forall F \in \mathcal{F},\ \forall (i,j) \in F \tag{3.23}
$$

Those constraints ensure that in a scenario F where an arc $(i,j) \in F$, we can route some flow through this arc (i,j) only if this arc is protected. Again, we can use the same columns-and-constraints generation algorithm as in Subsection 3.2.2, in order to find the most vital arcs in the separation problem among the non-fictive and non-protected arcs (we consider only combinations of selected but non-protected arcs when computing the set of maximum flows).

3.5 Valid and strengthening inequalities

In this section, we propose some valid or strengthening inequalities for both formulations to enhance the quality of the lower bound obtained by solving the continuous relaxation.

We first consider the case where we are not allowed to protect some arcs of the network. Secondly, we propose modifications of those inequalities to take the possibility of protecting arcs into account.

3.5.1 Case without the possibility of protecting arcs

Inequalities (3.24a) ensure that there are at least $k+1$ arcs entering each terminal. Indeed, if there are less than $k+1$ arcs entering it, then it is possible to delete all of them and thus to prevent one unit of flow from reaching the sink. Inequality (3.24b) states the same constraint for the arcs leaving the root.

$$\sum_{(i,t)\in A} y_{it} \quad \geq \quad k+1 \qquad \forall t \in T \tag{3.24a}$$

$$\sum_{(r,i)\in A} y_{ri} \quad \geq \quad k+1 \tag{3.24b}$$

Both Inequalities (3.24a) and (3.24b) are valid and cut some non-integer solutions. In Figure 3.2, we have $T = \{t_1\}$ and let u_i be the capacity of a_i for $i = 1, .., 5$ and y_i be the variable associated with the selection of a_i. The constraints associated with this graph for the cut-set formulation with $k = 1$ are $u_1y_1 \geq 1$, $u_2y_2 \geq 1$, $u_3y_3 \geq 1$, $u_4y_4 \geq 1$ and $y_5 = 1$. If we set $u_i = 2$ for all $i = 1, .., 4$ ($u_5 = 1$ because a_5 is a fictive arc), we have that the solution in which $y_1 = y_2 = y_3 = y_4 = 0.5$ and $y_5 = 1$ is optimal for the continuous relaxation since we consider positive costs in the objective function. Inequalities (3.24a) are then violated and impose that $y_3 + y_4 \geq 2$ (i.e. $y_3 = y_4 = 1$). Similarly, Inequalities (3.24b) impose that $y_1 + y_2 \geq 2$ (i.e. $y_1 = y_2 = 1$). In this case, the addition of both inequalities results in the cut of non-integer solutions (here, it even results in an optimal value of the integer problem equal to the optimal value of the continuous relaxation).

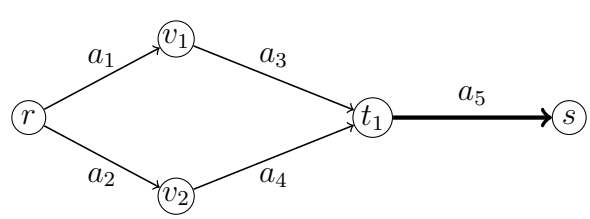


Figure 3.2 Graph where Inequalities (3.24a) and (3.24b) cut some non-integer solutions

Inequalities (3.25a) state that, for each Steiner vertex j, if an arc entering j is selected, then at least one arc leaving j must be selected, since all arc costs are assumed to be positive.

Inequalities (3.25b) state the same for arcs leaving a Steiner vertex j. Notice that these inequalities cut some integer but non-optimal solutions.

$$y_{ij} \leq \sum_{k \in \Gamma^+(j)} y_{jk} \qquad \forall j \in V \setminus \{T \cup \{r\}\},\ \forall i \in \Gamma^-(j) \tag{3.25a}$$

$$y_{jk} \leq \sum_{i \in \Gamma^-(j)} y_{ij} \qquad \forall j \in V \setminus \{T \cup \{r\}\},\ \forall k \in \Gamma^+(j) \tag{3.25b}$$

3.5.2 Case with the possibility of protecting arcs

The two families of Inequalities (3.24a) and (3.24b) are only true for the case without protection ($k' = 0$). Since one arc may be sufficient to ensure that one of the terminals is not isolated if it is protected, we can replace Inequalities (3.24a) and (3.24b) by (3.26a-3.26b) and (3.26c-3.26d) in this case. Inequalities (3.26a-3.26b) state that, for any terminal t, if there are no protected arcs entering t, there must be at least $k+1$ arcs entering t, otherwise there must be at least one. Inequalities (3.26c-3.26d) state the same constraint for the arcs entering the root.

$$\sum_{(i,t) \in A} y_{it} \geq 1 + (k(1 - \sum_{(i,t) \in A} p_{it})) \qquad \forall t \in T \tag{3.26a}$$

$$\sum_{(i,t) \in A} y_{it} \geq 1 \qquad \forall t \in T \tag{3.26b}$$

$$\sum_{(r,i) \in A} y_{ri} \geq 1 + (k(1 - \sum_{(r,i) \in A} p_{ri})) \tag{3.26c}$$

$$\sum_{(r,i) \in A} y_{ri} \geq 1 \tag{3.26d}$$

Inequalities (3.27) state that at least one arc entering a terminal t must be protected if there are less than $k+1$ arcs entering t.

$$\sum_{(i,t) \in A} p_{it} \geq 1 \qquad \forall t \in T \text{ with } |\Gamma^-(t)| \leq k \tag{3.27}$$

Inequalities (3.25a) and (3.25b) are still valid inequalities for the case with the possibility of protecting arcs.

3.6 Results analysis

In this section, we present the results of the tests for the formulations proposed previously. All experiments were performed on a computer with a 2.40GHz Intel(R) Core(TM) i7-5500U CPU and a 16GB RAM, using the solver CPLEX version 12.6.1, interfaced with Julia 0.6.0. We used in particular the package *JuMP*, a tool allowing mathematical modeling. For each test, the algorithm has been stopped after 3000 seconds if it has not terminated yet.

Table 3.1 Instance parameters

I	|V|	|T|	|E|
1	20	4	46
2	20	6	45
3	20	12	46
4	20	19	46
5	25	5	59
6	25	8	61
7	25	15	61
8	25	24	59
9	30	3	74
10	30	6	73
11	30	9	74
12	30	18	74
13	30	29	74
14	35	4	89
15	35	7	87
16	35	10	91
17	35	21	89
18	35	34	88
19	40	4	104
20	40	8	103

I	|V|	|T|	|E|
21	40	12	100
22	40	24	104
23	40	39	103
24	45	4	119
25	45	9	118
26	45	14	118
27	45	27	114
28	45	44	119
29	50	5	133
30	50	10	133
31	50	15	131
32	50	30	133
33	50	49	130
34	60	6	160
35	60	12	157
36	60	18	161
37	60	36	161
38	60	59	161
39	70	7	188
40	70	14	189

I	|V|	|T|	|E|
41	70	21	188
42	70	42	190
43	70	69	189
44	80	8	221
45	80	16	219
46	80	24	216
47	80	48	213
48	80	79	223
49	90	9	249
50	90	18	248
51	90	27	250
52	90	54	248
53	90	89	247
54	100	10	281
55	100	20	279
56	100	30	278
57	100	60	277
58	100	99	282

Table 3.1 shows for each instance its number I, as well as the number of vertices $|V|$, terminals $|T|$, and edges $|E|$. All instances have been generated in the following way: the vertices have been randomly generated in the plane, and the capacity of an arc is more likely to be high if this arc is close to the root. We compute 5 instances for each different value of $|V|$ selected (except for $|V| = 20$ and $|V| = 25$ because the number of vertices is too small): for each one we assign a different value of $|T|$ selected in $\{\frac{1}{10}|V|, \frac{2}{10}|V|, \frac{3}{10}|V|, \frac{6}{10}|V|, |V|-1\}$. The graph associated with each instance is parse, corresponding to the wind-farm application where graphs are not dense. Also, the formulations are very dependent on the number of arcs and graphs which are too dense would be too difficult to solve (for similar reasons, the number of vertices is bounded by 100). The arc capacities are high enough to ensure that

there is at least one feasible solution to our problem, but low enough to keep the problem difficult enough to solve. More precisely, the capacities are chosen randomly among four values: $0.8|T|$, $0.6|T|$ and, except for the edges with endpoints at distance 1 or 2 from the root, $0.4|T|$ and $0.2|T|$. Furthermore, the cost of an arc depends on both its length and its capacity, and hence is not necessarily integral.

Table 3.2 presents the results for the cutset and flow formulations for one possible arc deletion and no protection allowed (i.e. $k = 1$ and $k' = 0$) on instances with non-uniform capacities. The column I gives the number of the instance on which the formulation is tested, the column gap_{LR} gives the gap between the optimal value of the initial continuous relaxation with the valid inequalities and the optimal integer value (please remind that the relaxations of the flow and cutset formulations are equal, see Theorem 3.3.2). For each formulations, the column gap_f gives the final gap between the best lower bound $best_{LB}$ and the best integer solution found $best_I$ (if the gap is equal to 0, we have found an optimal solution, otherwise we give the gap obtained after 3000 seconds of computation): formally we have that $gap_f = (best_I - best_{LB})/best_I$; the column *time* gives the time in seconds needed to find an optimal solution or 3000 if the algorithm has not terminated yet; the column *it* gives the number of iterations performed by the algorithm (i.e. the number of times the algorithm of constraints generation is executed); the column *nodes* gives the number of nodes explored in the branch-and-cut.

In Table 3.2, we can observe that both formulations manage to find an optimal solution to the problem within 3000 seconds for instances with a number of vertices smaller than or equal to 45, except for two instances for the flow formulation. However, the cutset formulation has a better solving time for almost all instances, although it performs more iterations of constraints generation and has a higher number of nodes explored in the branch-and-cut. This can be explained by the fact that although the number of constraints that can be generated in the cutset formulation is importantly higher than the family of constraints that can be generated in the flow formulation for $k = 1$ (whose number is bounded by $|E|$), at each constraint generation in the flow formulation, we add at least $|E|$ variables to the formulation. On instances having 50 to 100 vertices, the cutset formulation allows to find a feasible solution which is often optimal or has a value close to the optimal one. The algorithm associated with the cutset formulation finds the optimal solution for 39 instances out of 52 tested. For the other instances, the mean final gap is equal to 0.07. The algorithm associated with the flow formulation finds the optimal solutions for 27 instances out of 38 tested (all those instances have been optimally solved by the cutset formulation except Instance 38 which presents a

Table 3.2 Results for non-uniform capacities, $k = 1$ and $k' = 0$

		Cutset				Flow			
I	gap_{LR}	gap_f	time (s)	it	nodes	gap_f	time (s)	it	nodes
1	0.15	0	0.4	13	0	0	0.1	2	32
2	0.25	0	1.7	46	800	0	0.9	13	279
3	0.28	0	5.8	113	4859	0	6.3	15	2237
4	0.18	0	4.2	92	4423	0	6.2	20	1995
5	0.19	0	3.0	61	554	0	1.4	14	273
6	0.16	0	3.6	79	5194	0	9.5	12	2426
7	0.24	0	8.8	127	19143	0	28.6	30	2290
8	0.1	0	0.4	13	179	0	0.5	7	35
9	0.2	0	5.7	106	4141	0	28.6	24	866
10	0.21	0	8.8	126	11361	0	64.6	23	8408
11	0.17	0	8.5	115	13161	0	41.3	16	7995
12	0.07	0	0.5	13	13	0	0.38	4	13
13	0.28	0	18.1	243	18441	0	87.4	26	3488
14	0.25	0	10.0	143	8188	0	23.2	13	4070
15	0.1	0	5.5	67	552	0	8.9	17	1316
16	0.22	0	6.7	129	4318	0	20.8	15	1540
17	0.17	0	6.4	93	3586	0	41.2	24	2961
18	0.26	0	710	391	15.10^4	0	1947	34	35683
19	0.15	0	13.6	116	20339	0	251	37	5016
20	0.08	0	10.7	98	1964	0	9.16	5	1432
21	0.16	0	3.4	70	601	0	4.9	11	363
22	0.18	0	32.4	217	50927	0	384	16	28156
23	0.19	0	19.6	149	21588	0	466	22	7129
24	0.24	0	2050	766	57.10^4	0.09	3000	36	21610
25	0.13	0	31.1	182	62060	0	353	20	22834
26	0.28	0	21.2	188	24570	0	1642	27	8680
27	0.21	0	1335	533	19.10^4	0.1	3000	34	8513
28	0.29	0	20.0	155	22125	0	1812	29	12645
29	0.08	0	8.6	66	1880	0	159.7	21	1431
30	0.34	0	2950	557	14.10^4	0.3	3000	45	2015
31	0.19	0	300.7	319	26.10^3	0.05	3000	29	32515
32	0.24	0	130.0	301	16.10^3	0.09	3000	44	7547
33	0.16	0	127.8	302	21.10^3	0.03	3000	38	5995
34	0.33	0	223.6	369	20.10^3	0.15	3000	51	4065
35	0.27	0	1102	667	10.10^4	0.26	3000	56	937
36	0.17	0	73.1	304	65183	0.12	3000	49	2057
37	0.16	0	791.4	386	13.10^4	0.12	3000	50	2234
38	0.13	0.02	3000	580	99.10^4	0.13	3000	55	1458
39	0.18	0	245.3	436	16.10^3	-	-	-	-
40	0.27	0.09	3000	939	19.10^4	-	-	-	-
41	0.17	0.03	3000	1050	35.10^4	-	-	-	-
42	0.1	0.02	3000	827	59.10^4	-	-	-	-
43	0.32	0.18	3000	928	11.10^4	-	-	-	-
44	0.25	0.11	3000	957	11.10^4	-	-	-	-
45	0.26	0.15	3000	1142	12.10^4	-	-	-	-
46	0.17	0.08	3000	1227	30.10^4	-	-	-	-
47	0.1	0.01	3000	799	77.10^4	-	-	-	-
48	0.27	0.1	3000	1155	10.10^4	-	-	-	-
49	0.26	0.1	3000	909	10.10^4	-	-	-	-
50	0.26	0.12	3000	1190	12.10^4	-	-	-	-
51	0.13	0	964.2	558	89.10^3	-	-	-	-
52	0.12	0.01	3000	672	80.10^4	-	-	-	-

final gap equal to 0.02). For the other instances, the mean final gap is equal to 0.13.

Table 3.3 Results for non-uniform capacities, $k = 2$ and $k' = 0$

		Cutset				Flow			
I	gap_{LR}	gap_f	time (s)	it	nodes	gap_f	time (s)	it	nodes
1	0.06	0.0	0.6	23	35	0.0	0.6	3	5
2	0.15	0.0	1.2	27	104	0.0	4.7	22	377
6	0.13	0.0	4.4	90	5405	0.0	172	55	5110
7	0.17	0.0	6.0	106	3602	0.0	196	52	5968
11	0.15	0.0	6.4	97	3150	0.0	664	83	12249
16	0.18	0.0	11.7	161	3989	0.0	878	75	9969
19	0.22	0.0	22.5	291	16159	0.0	2710	74	14376
21	0.22	0.0	160	495	332748	0.13	3000	94	14827
23	0.04	0.0	5.4	40	435	0.0	194	18	1621
24	0.11	0.0	6.0	123	1843	-	-	-	-
25	0.12	0.0	25.6	211	32959	-	-	-	-
26	0.12	0.0	24.2	216	15778	-	-	-	-
29	0.24	0.0	59.3	267	72991	-	-	-	-
30	0.16	0.0	1085	582	2039248	-	-	-	-
31	0.21	0.0	79.4	250	152911	-	-	-	-
34	0.28	0.07	3000	1339	1524965	-	-	-	-
36	0.17	0.0	229	406	249593	-	-	-	-
37	0.12	0.0	52.6	202	54918	-	-	-	-
39	0.25	0.0	193	541	143432	-	-	-	-
40	0.18	0.0	966	650	944465	-	-	-	-
44	0.22	0.04	3000	945	1236272	-	-	-	-
49	0.26	0.15	3000	1542	1027765	-	-	-	-
50	0.18	0.08	3000	966	1750500	-	-	-	-
51	0.2	0.1	3000	1536	2051800	-	-	-	-
52	0.13	0.06	3000	1232	3777198	-	-	-	-
53	0.04	0.0	37.7	70	7970	-	-	-	-
54	0.27	0.13	3000	1717	1097985	-	-	-	-
55	0.25	0.14	3000	1670	1156704	-	-	-	-
57	0.09	0.0	149	257	66364	-	-	-	-
58	0.03	0.0	49.3	95	4253	-	-	-	-

Table 3.3 presents the results of the cutset and flow formulations for $k = 2$ and $k' = 0$ on instances with non-uniform capacities. The columns are defined similarly to the ones in Table 3.2. For some instances, the instance does not admit any feasible solution for $k = 2$ and $k' = 0$, thus we remove from the table those instances (the formulations determine when no solution exists within a few seconds for this set of instances). Again, the cutset formulation is way more efficient than the flow formulation. The flow formulation still presents a smaller number of constraints generation iterations but the difference with the number of iterations of the cutset formulation is less important than for $k = 1$, due to the fact that we now may

Table 3.4 Results for non-uniform capacities, $k = 3$ and $k' = 0$

		Cutset				Flow			
I	gap_{LR}	gap_f	time (s)	it	nodes	gap_f	time (s)	it	nodes
6	0.1	0.0	6.5	121	1409	0.0	1209	68	4326
7	0.06	0.0	3.0	29	136	0.0	98.4	19	255
16	0.12	0.0	88.1	78	1041	0.04	3000	69	1365
19	0.21	0.0	63.5	380	82935	-	-	-	-
25	0.08	0.0	10.2	130	2526	-	-	-	-
34	0.27	0.12	3000	1474	1972379	-	-	-	-
39	0.26	0.03	3000	1028	2651353	-	-	-	-
40	0.19	0.0	1194	770	1578225	-	-	-	-

have to generate up to $\binom{|E|}{2}$ constraints, whereas the number of iterations for the cutset formulations is only slightly higher for $k = 2$ than for $k = 1$.

In Table 3.3, the cutset formulation manages to find an optimal solution for most of the instances of at most 70 vertices, and does not appear to be as sensitive to the value of k as the flow formulation. For instances with 80 vertices and more, the cutset formulation manages to find a feasible solution which is optimal or has a value reasonably close to the optimal one. The algorithm associated with the cutset formulation finds the optimal solution for 22 instances out of 30 tested. For the other instances, the mean final gap is equal to 0.9.

Table 3.4 presents the results of the cutset and flow formulations for $k = 3$ and $k' = 0$ on instances with non-uniform capacities. The columns are defined similarly to the previous tables. Again, some instances do not admit any feasible solution for $k = 3$ and $k' = 0$, thus we remove from the table those instances. Similarly to the previous results for $k \in \{1, 2\}$, the cutset formulation is more efficient. Furthermore, the number of constraints generated in the flow formulation is closer to the one of the cutset formulation than in $k \in \{1, 2\}$.

Tables 3.5, 3.6 and 3.7 present results for our problem with the possibility of protecting some arcs. In real applications, protecting an arc can correspond to double the cable or adding devices protecting the cables against the environment (like extreme cold). Since those modifications can be expensive, we only consider small values of k'.

Table 3.5 presents the results of the cutset and flow formulations for $k = 1$ and $k' \in \{1, 2, 3\}$ on instances with non-uniform capacities. The column I gives the set of the instances on

Table 3.5 Results for non-uniform capacities, $k = 1$ and $k' \in \{1, 2, 3\}$

			Cutset				Flow			
I	k'	$\overline{gap_{LR}}$	$\overline{gap_f}$	$\overline{time(s)}$	$\overline{it}$	$\overline{n}$	$\overline{gap_f}$	$\overline{time}$ (s)	$\overline{it}$	$\overline{n}$
1-10	1	0.24	0.0	5.0	92	6941	0.0	25.9	17	2826
-	2	0.27	0.0	7.9	114	12897	0.0	59.1	29	3017
-	3	0.3	0.0	11.1	125	20934	0.0	66.4	31	4052
11-20	1	0.23	0.0	12.0	141	13319	0.0	201	35	3734
-	2	0.25	0.0	16.0	158	19726	0.0	337	41	5099
-	3	0.28	0.0	24.2	182	40042	0.0	642	52	7656
21-30	1	0.22	0.01	727	346	8.10^4	0.07	2308	38	11728
-	2	0.24	0.01	937	388	9.10^4	0.06	2290	47	7198
-	3	0.26	0.02	945	424	9.10^4	0.08	2561	55	8795
31-40	1	0.26	0.01	952	402	5.10^4	0.19	2740	46	3764
-	2	0.28	0.01	1207	494	7.10^4	0.23	3000	58	3514
-	3	0.3	0.02	1321	531	6.10^4	0.23	3000	62	2532
41-50	1	0.24	0.08	2176	818	11.10^4	-	-	-	-
-	2	0.27	0.1	2590	932	13.10^4	-	-	-	-
-	3	0.29	0.13	2755	1028	14.10^4	-	-	-	-
51-58	1	0.2	0.1	3000	1237	17.10^4	-	-	-	-
-	2	0.23	0.14	3000	1564	14.10^4	-	-	-	-
-	3	0.24	0.15	3000	1466	15.10^4	-	-	-	-

Table 3.6 Results for non-uniform capacities, $k = 2$ and $k' \in \{1, 2, 3\}$

			Cutset				Flow			
I	k'	$\overline{gap_{LR}}$	$\overline{gap_f}$	$\overline{time}$ (s)	$\overline{it}$	$\overline{n}$	$\overline{gap_f}$	$\overline{time}$ (s)	$\overline{it}$	$\overline{n}$
1-10	1	0.14	0.0	7.5	108	9088	0.0	384	48	7971
-	2	0.2	0.0	14.6	152	21575	0.02	1047	64	10902
-	3	0.22	0.0	28.2	190	53192	0.05	1238	78	14521
11-20	1	0.19	0.0	18.7	167	17192	0.04	1818	69	11621
-	2	0.22	0.0	37.7	261	42657	0.12	2598	86	14564
-	3	0.24	0.0	80.4	291	100650	0.15	2533	98	12030
21-30	1	0.17	0.0	450	397	590660	-	-	-	-
-	2	0.19	0.0	606	491	829507	-	-	-	-
-	3	0.22	0.02	1066	531	1148662	-	-	-	-
31-40	1	0.23	0.02	1275	699	856019	-	-	-	-
-	2	0.26	0.06	1514	746	726194	-	-	-	-
-	3	0.28	0.08	1592	797	796008	-	-	-	-
41-50	1	0.2	0.08	1719	908	1298931	-	-	-	-
-	2	0.23	0.1	1909	1113	902446	-	-	-	-
-	3	0.3	0.16	2503	1367	1295592	-	-	-	-
51-58	1	0.23	0.05	1842	929	733356	-	-	-	-
-	2	0.24	0.07	1924	1209	944623	-	-	-	-
-	3	0.3	0.1	2632	1406	1343762	-	-	-	-

which the formulation is tested, the column $\overline{gap_{LR}}$ gives the mean value of the gap between the optimal value of the continuous relaxation and the optimal value of the integer formu-

lation for each instance of the data set tested. For each formulation, the column $\overline{gap_f}$ gives the mean value of the final gap of each instance of the data sets between the best lower bound and the best integer solution found (if the gap is equal to 0, we have found an optimal solution, otherwise we give the gap obtained after 3000 seconds of computation); the column $\overline{time}$ gives the mean solving time to find an optimal solution or 3000 if the algorithm has not terminated yet for the data set; the column $\overline{it}$ gives the mean number of iterations performed by the algorithm (i.e. the number of times the algorithm of constraints generation is executed) for the data set; the column $\overline{n}$ gives the mean number of nodes explored in the branch-and-cut for the data set.

One can see that the cutset formulation is also more efficient with the addition of the possibility of protecting arcs. The solving time is higher in the case with $k' > 0$ than in the case with $k' = 0$ and it increases as well as the gaps and the number of iterations and nodes when k' becomes bigger. It appears that the flow formulation is slightly less impacted by the variation of k'.

Table 3.6 presents the results of the cutset and flow formulations for $k = 2$ and $k' \in \{1, 2, 3\}$ on instances with non-uniform capacities; the columns are similar to the ones in Table 3.5. The flow formulation is logically still importantly slower when k increases in the protected case, and the mean number of iterations is around the double of the one for $k = 1$. Again, the cutset formulation appears to be less impacted by the increase of k in the protected case: the gaps are equivalent, the solving times appear to be even better for some sets of instances (sets 41-50 and 51-58), and the number of iterations does not increase as much as the ones of the flow formulation.

Table 3.7 presents the results of the cutset and flow formulations for $k = 3$ and $k' \in \{1, 2, 3\}$ on instances with non-uniform capacities; the columns are similar to the ones in Table 3.5. Similar remarks as the ones of Table 3.6 can be made: the flow formulation is much slower when $k = 3$ whereas the cutset formulation is not as much impacted by the incrementation of k.

Table 3.8 presents the results of the cutset formulations for $k = 1$ and $k' = 0$ on instances with uniform capacities. We compute the tests on the *Regular-Cutset* formulation (which corresponds to the classic cutset formulation proposed in Equation (3.2) in Subsection 3.2.1) and the *Uniform-Cutset* formulation (which corresponds to the modification of the formula-

Table 3.7 Results for non-uniform capacities, $k = 3$ and $k' \in \{1, 2, 3\}$

			Cutset				Flow			
I	k'	$\overline{gap_{LR}}$	$\overline{gap_f}$	$\overline{time}$	$\overline{it}$	$\overline{n}$	$\overline{gap_f}$	$\overline{time}$	$\overline{it}$	$\overline{n}$
1-10	1	0.14	0.0	16.7	234	9584	0.08	1835	114	4384
-	2	0.21	0.0	16.6	186	25975	0.06	1910	132	3757
-	3	0.23	0.0	29.9	221	53703	0.15	1992	136	3948
11-20	1	0.21	0.0	117	372	108708	0.13	3000	66	588
-	2	0.21	0.0	234	359	247729	0.18	3000	90	914
-	3	0.3	0.01	472	503	340496	0.25	2869	76	1413
21-30	1	0.16	0.0	149	364	272985	-	-	-	-
-	2	0.28	0.02	1078	715	1182379	-	-	-	-
-	3	0.3	0.03	990	827	678130	-	-	-	-
31-40	1	0.31	0.07	1843	1127	1038762	-	-	-	-
-	2	0.36	0.11	2021	1392	994130	-	-	-	-
-	3	0.37	0.14	2049	1512	975210	-	-	-	-
41-50	1	0.31	0.08	3000	1580	1950209	-	-	-	-
-	2	0.27	0.11	2430	1683	824006	-	-	-	-
-	3	0.35	0.21	2451	2246	680713	-	-	-	-
51-58	2	0.26	0.0	1174	350	1189524	-	-	-	-
-	3	0.36	0.0	2150	674	1156240	-	-	-	-

tion to reduce the number of constraints in the case of uniform capacities, see Equation (3.5) in Subsection 3.2.1). For each instance, we test the formulations when uniform capacities are equal to $0.4|T|$, $0.6|T|$ and $0.8|T|$. Column I gives the instance on which the formulations are tested; column $\overline{time}$ gives the solving time of the instance for each formulation; column $\overline{it}$ gives the number of iterations of the constraints generation algorithm for the instance for each formulation; column $\overline{nb_n}$ gives the number of nodes generated for the instance for each formulation. We did not give the final gap since each instance has been optimally solved by both formulations within the 3000 seconds given to the algorithms.

The results are impressive in comparison to the case with non-uniform capacities. The solving times are much smaller, as well as the mean number of nodes and the mean number of iterations for both *Regular-Cutset* and *Uniform-Cutset* formulations. The *Uniform-Cutset* formulation appears to be more efficient, which appears to be logical since the number of constraints of the formulation is highly reduced and the number of variables remains the same. The number of iterations is always smaller than the one of the *Regular-Cutset* formulation and the number of nodes is often smaller than the one of the regular formulation too.

Table 3.9 presents the results obtained with the *Regular-Cutset* and *Uniform-Cutset* formulations for $k = 2$ and $k' = 0$ on instances with uniform capacities. Columns are defined as

Table 3.8 Results for uniform capacities and $k = 1$

	Uniform-Cutset			Regular-Cutset		
I	$\overline{time}(s)$	$\overline{it}$	$\overline{nb_n}$	$\overline{time}(s)$	$\overline{it}$	$\overline{nb_n}$
1	0.07	2	0	0.11	4	0
2	0.28	6	2	0.65	28	184
3	4.98	32	1215	5.39	44	978
4	0.24	5	0	0.32	9	2
5	0.36	8	0	0.9	29	285
6	0.44	10	46	0.64	22	146
7	0.43	8	0	1.1	28	138
8	0.28	3	0	0.37	7	0
9	0.25	5	28	0.31	11	67
10	0.59	14	23	1.02	32	234
11	0.39	8	0	0.89	24	111
12	0.61	11	21	1.44	36	399
13	0.46	7	1	0.87	16	44
14	0.23	5	0	0.34	10	15
15	0.35	5	0	0.51	15	57
16	0.54	9	32	1.96	45	445
17	0.74	10	0	2.01	37	254
18	0.36	4	0	0.33	5	0
19	0.53	8	208	1.11	30	539
20	0.36	5	3	0.6	13	22
21	0.72	10	55	1.47	28	608
22	1.2	14	124	2.78	41	691
23	0.55	5	2	0.82	11	21
24	0.36	5	2	0.57	15	69
25	0.7	8	118	1.12	19	270
26	1.62	17	276	3.58	49	937
27	2.55	24	146	7.19	81	1606
28	1.42	11	4	1.83	17	23
29	1.39	17	589	3.34	57	3329
30	2.07	21	853	5.4	71	2904
31	1.2	12	12	4.81	59	2174
32	0.54	4	0	0.87	8	0
33	2.65	15	33	4.3	29	193
34	3.25	33	1009	8.79	86	9662
35	1.42	13	185	2.07	24	274
36	3.78	31	642	10.37	95	5146
37	3.32	19	111	7.2	48	1967
38	2.38	9	50	3.41	17	58
39	2.98	27	1273	7.8	88	4958
40	3.14	25	406	6.57	60	2351
41	3.22	22	709	6.82	60	1011
42	1.27	5	19	2.74	15	114
43	3.76	12	112	12.69	52	482
44	6.39	43	2325	25.06	140	27009
45	5.85	40	1772	13.56	92	8873
46	7.76	36	1409	16.07	84	6268
47	6.32	25	217	10.25	45	1034
48	8.47	26	481	34.09	113	2946
49	11.72	56	5350	76.47	134	85820
50	10.18	37	1244	24.34	127	9320
51	12.99	51	2975	65.37	181	53467
52	15.06	40	1247	33.08	99	5027
53	13.43	20	198	26.14	46	646

Table 3.9 Results for uniform capacities and $k = 2$

	Uniform-Cutset			Regular-Cutset		
I	$\overline{time}(s)$	$\overline{it}$	$\overline{nb_n}$	$\overline{time}(s)$	$\overline{it}$	$\overline{nb_n}$
1	0.31	7	5	0.65	20	57
2	0.33	8	6	1.27	44	405
5	0.49	10	61	2.02	56	585
6	0.84	16	134	2.69	68	923
7	0.77	11	10	1.51	34	265
9	0.28	4	2	1.14	30	134
10	1.44	26	111	5.25	106	2122
11	0.82	14	0	3.58	71	1027
16	0.68	9	19	2.94	52	418
19	1.44	22	859	6.85	116	7179
20	3.25	34	1310	8.77	108	4528
21	2.6	31	463	8.97	121	4061
22	2.19	16	99	5.06	47	639
23	0.73	4	1	0.85	8	2
24	0.97	16	85	4.4	85	1996
25	3.84	43	567	9.56	120	4451
26	2.59	22	228	11.56	111	3082
28	0.85	4	0	0.77	6	0
29	3.86	42	2783	37.0	260	45720
30	7.09	68	2328	33.25	236	33383
31	4.73	39	1338	38.18	233	48585
32	1.55	7	24	1.59	11	2
33	0.41	1	0	0.36	2	0
34	30.24	85	43667	2720.92	645	3252043
36	6.79	42	2504	34.65	189	28947
37	4.27	19	150	28.48	132	3862
38	1.96	6	0	2.59	10	0
39	4.95	43	1789	40.9	276	35710
40	14.27	78	7095	198.69	344	284762
41	6.79	36	505	16.62	102	1704
42	8.03	27	1210	25.26	97	3832
43	3.81	9	94	5.0	15	42
44	50.38	118	73169	2656.92	744	1733895
49	68.69	110	65440	1536.17	421	1310976
50	59.39	108	90681	1450.07	460	1657034
51	46.41	108	27507	2678.41	533	3052695
52	37.85	67	6070	222.36	270	341739
53	17.27	23	1214	30.83	55	2869

in Table 3.8. For $k = 2$, the uniform formulation appears to be way more efficient than the regular one. The mean solving time and number of iterations and nodes have only slightly

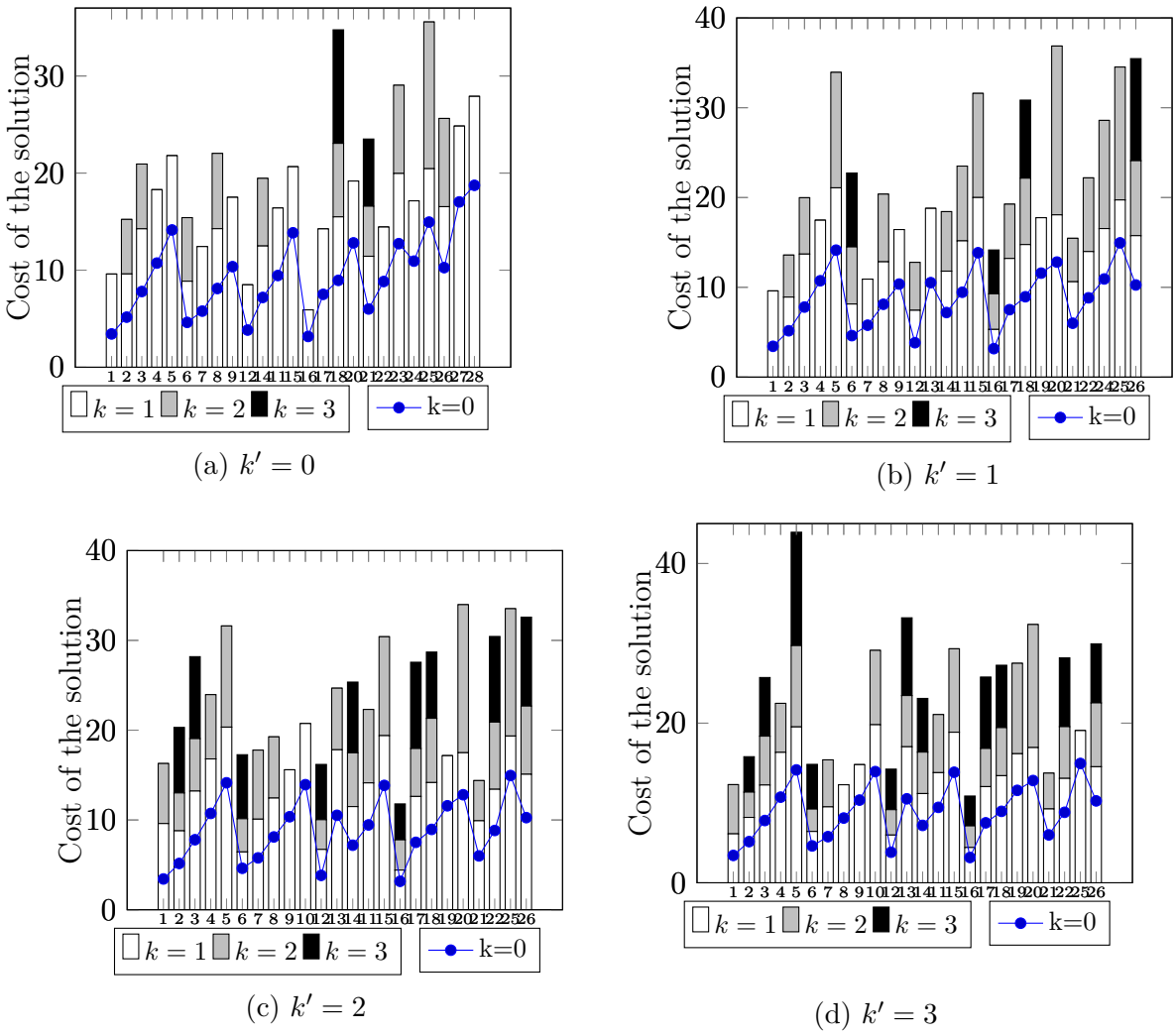
increased with the incrementation of k for the uniform formulation (the mean solving time for each instance is bounded by 70 seconds) whereas it has consequently increased for the regular one (the mean solving time is around 2678 seconds for Instance 51 for example).

Table 3.10 Results for uniform capacities and $k = 3$

	Uniform-Cutset			Regular-Cutset		
I	$\overline{time}(s)$	$\overline{it}$	$\overline{nb_n}$	$\overline{time}(s)$	$\overline{it}$	$\overline{nb_n}$
1	0.19	5	0	0.46	13	0
6	0.74	9	18	1.72	36	184
7	0.43	3	0	0.66	8	0
9	0.66	7	0	2.42	53	711
16	0.77	9	5	4.32	58	495
19	1.88	21	220	8.51	115	3565
20	0.79	6	0	6.71	68	575
24	0.75	9	0	3.63	55	428
34	14.39	73	3669	594.06	635	720890
39	7.91	45	1692	71.66	349	48393
40	11.37	39	593	87.91	320	28594
44	16.17	66	1635	197.38	581	108704
49	38.66	76	5924	644.54	553	358451

Table 3.10 presents the results obtained with the *Regular-Cutset* and *Uniform-Cutset* formulations for $k = 3$ and $k' = 0$ on instances with uniform capacities. Columns are defined as in Tables 3.8 and 3.9. Again, the uniform formulation appears to be a consequent upgrade of the regular one, and the solving times are really smaller than in the non-uniform case.

We also compute the results obtained by both formulations, without the addition of the valid inequalities proposed in Subsection 3.5, on a subset of instances with $k \in \{1, 2, 3\}$, $k' = 0$, and non-uniform capacities. Let $\overline{\Delta}^C_{time}$ (respectively $\overline{\Delta}^F_{time}$) be the mean augmentation of the solving time when these valid inequalities are removed from the cut-set (respectively flow) formulation. On the test instances, $\overline{\Delta}^C_{time}$ is equal to 3.24 (meaning that the solving time is multiplied by 3.24 on average without the valid inequalities), while $\overline{\Delta}^F_{time}$ is equal to 10.78. Hence, adding these valid inequalities has a huge impact on the solving time, especially on the flow formulation. Furthermore, let $\overline{\Delta}_{CR}$ be the mean augmentation of the optimal value of the continuous relaxation when these valid inequalities are added to the formulation. On the test instances, $\overline{\Delta}_{CR}$ is equal to 1.28 (meaning that the optimal value of the continuous relaxation is multiplied by 1.28 on average with these valid inequalities). The optimal value of the continuous relaxation is then consequently increased when we add these valid inequalities.



(a) $k' = 0$

(b) $k' = 1$

(c) $k' = 2$

(d) $k' = 3$

Figure 3.3 Cost of the solutions for different instances

Figure 3.3 deals with the cost of designing failure-resilient networks; the number of the corresponding test instance is displayed on the x-axis. Each subfigure shows the cost of an optimal solution for the case where k equals 0 (no arc deleted), 1, 2 and 3. The subfigure (a) corresponds to the case where $k' = 0$ (no protection allowed), whereas (b), (c) and (d) correspond to the case where k' is equal to 1, 2 and 3, respectively. This figure shows that designing a network resilient to even a small number of arc-failures can be costly (the cost increases greatly with the value of k). However, on subfigure (d), we can see that, by protecting a sufficiently large but still small subset of arcs on the test instances, one can obtain networks that are resilient to 1 or 2 arc deletions while maintaining a cost close to the optimal value of the case with no arc failures.

CHAPTER 4 A TABU SEARCH FOR THE DESIGN OF CAPACITATED ROOTED k-EDGE CONNECTED STEINER PLANAR NETWORKS

4.1 Introduction

In this chapter, we focus on the design of networks which are resilient to one or several breakdowns, studying the Capacitated Rooted k-Edge Connected Steiner Network Problem (CRkECSN) defined in Chapter 3 in the case of planar networks. We recall that $G = (V, A)$ is a directed input graph, that c and u are respectively a cost and a capacity function on A, and that r and $|T|$ are respectively the root and the terminals. We say that G has a feasible flow if it is possible to route one unit of flow from the root to each terminal while respecting the arc capacity constraints. Given an integer $k \geq 0$, a subgraph $G' = (V, A')$ of G is said to be *k-survivable* if every subgraph obtained from G' by removing at most k arcs of A' has a feasible flow. We aim at selecting a minimum cost subset $A' \subseteq A$ of arcs such that $G' = (V, A')$ is k-survivable.

In this chapter, we focus on planar graphs which are useful in practice since many underlying graphs in real-world networks are planar. In the next section, we describe a procedure that determines in polynomial time whether a given planar graph is k-survivable. In Section 4.3, we embed this procedure into a tabu search. Computational experiments and comparisons with the exact algorithm introduced in Chapter 3 are presented in Section 4.4.

4.2 Testing survivability

Given a subset A' of the original arc set A, we present in this section a method to test whether $G' = (V, A')$ is k-survivable, based on a method proposed in [68]. Every arc $a = (i, j) \in A'$ has an associated upper bound $u_a = u_{ij}$ (called capacity) on the amount of flow that can traverse it, while the lower bound $l_a = l_{ij}$ is equal to zero since there is no imposed flow on any arc of G' (we introduce lower bounds here because they will be used later for other arcs).

The first step of the procedure consists in determining a subset of arcs $\mathcal{S}$ in the complete graph associated with V such that $\mathcal{S}$ corresponds to a tree rooted at r spanning all terminals of T and $\mathcal{S}$ does not break the planarity of the input graph G' (i.e. if we add the arcs of $\mathcal{S}$ to G', the graph $G'' = (V, A' \cup \mathcal{S})$ is still planar), which is an easy task. Note that such a tree necessarily exists since r is a root and we do not forbid the graph $G'' = (V, A' \cup \mathcal{S})$ to

be a multigraph (i.e. we can select in $\mathcal{S}$ arcs from A'): an easy solution to find $\mathcal{S}$ is then to compute a tree using arcs of A' (and this is what we will do).

For every vertex v in $\mathcal{S}$, let n_v be the number of terminals in the subtree rooted at v. For each arc (i, j) in $\mathcal{S}$, we add the reverse arc (j, i) in G' and set $l_{ji} = u_{ji} = n_j$ (there is no issue if (j, i) is already in A' since the resulting graph can be a multigraph). We denote by A_R the set of arcs added to G' and by $\widetilde{G'}$ the resulting graph. An example is depicted in Figure 4.1. The left graph G' has two terminals t_1 and t_2. The directed tree $\mathcal{S}$ with arc set $\{(r, u), (u, t_1), (u, v), (v, t_2)\}$ induces the graph $\widetilde{G'}$ on the right, the arcs in A_R being represented by bold lines. Since $n_r = n_u = 2$ while $n_v = n_{t_1} = n_{t_2} = 1$, all arcs (j, i) in A_R have $l_{ji} = u_{ji} = 1$, except the arc (u, r) for which $l_{ur} = u_{ur} = 2$.

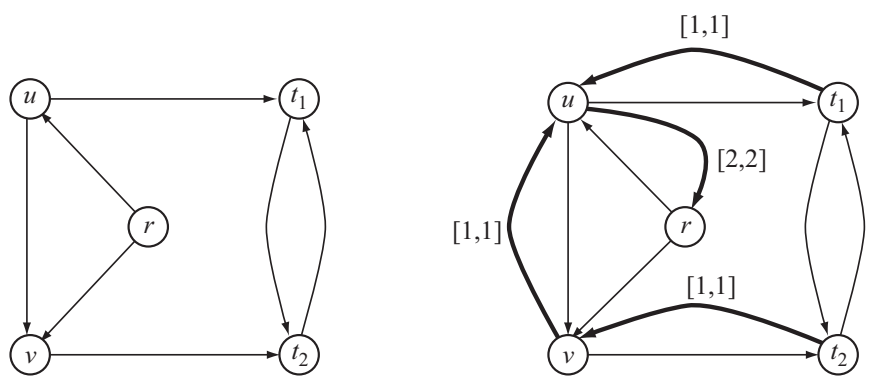


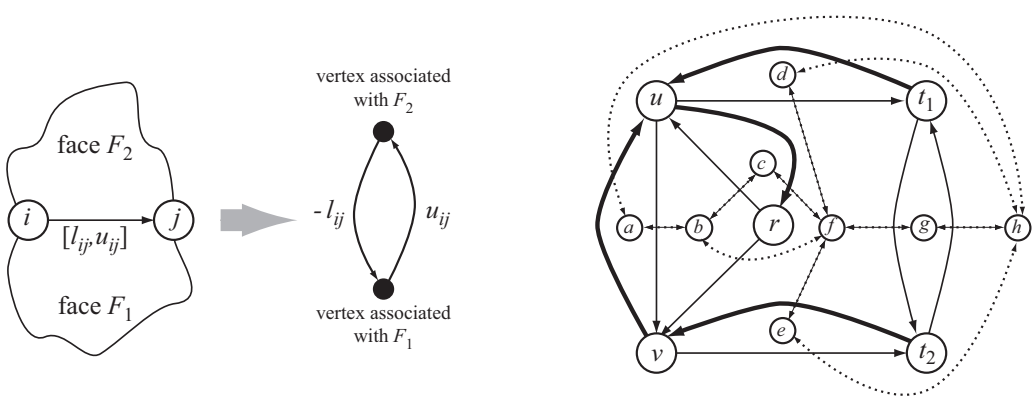
Figure 4.1 Construction of $\widetilde{G'}$ from G'

The circulation problem is a generalization of the network flow problem where flow conservation is required for all vertices (i.e. there are no source and sink). It is easy to observe that the original graph G' has a feasible flow routing a unit of flow from r to each terminal if and only if the extended graph $\widetilde{G'}$ has a feasible circulation: the flow sent from the root to the terminals travels back to r using the arcs in A_R.

The problem of determining whether G' is k-survivable was shown to be NP-complete in the general case [66], but can be solved in polynomial time if the graph $G' + t$ obtained from G' by adding a sink t to which all terminals are linked is planar [50]. Since the addition of a sink to a planar graph G' does not necessarily preserves planarity, Zenklusen [68] has shown how to solve the problem in polynomial time when G' (and not necessarily $G' + t$) is planar. His procedure is described here below.

Notice first that if G' is planar, then $\widetilde{G'}$ is also planar. We can therefore consider its dual. By convention (see [44] for more details), the dual of a directed planar graph is obtained by creating a vertex for each face of the original graph, and by connecting two vertices by an arc if they correspond to faces in the original graph sharing an arc. Every dual arc is oriented so that it corresponds to a 90° anticlockwise turn from the corresponding primal arc. As proposed by Zenklusen [68], we consider here an extended dual graph that contains not only the standard dual arcs, but also the arcs with the opposite direction. More precisely, we build a graph $D_{\widetilde{G'}}$ from $\widetilde{G'}$ as follows. We first create a vertex in $D_{\widetilde{G'}}$ for every face in $\widetilde{G'}$. Then, for every arc (i, j) in $\widetilde{G'}$, we consider the two faces F_1 and F_2 that (i, j) separates (see Remark 4.2.1), where F_1 is the face below (i, j) when (i, j) is drawn horizontally with an arrow going from left to right. We create an arc (F_1, F_2) of length u_{ij} and an arc (F_2, F_1) of length $-l_{ij}$ in $D_{\widetilde{G'}}$. Note that (F_1, F_2) is the standard dual arc associated with (i, j), while (F_2, F_1) is the opposite one. We also denote by a^d_{ij} the standard dual arc (F_1, F_2) in order to distinguish it from its opposite. Note that the arc (F_2, F_1) has length 0 if $(i, j) \in A'$ (since there is no imposed flow on any arc of G'), while its length is $-n_j$ if $(i, j) \in A_R$.

The construction of $D_{\widetilde{G'}}$ from $\widetilde{G'}$ is illustrated in Figure 4.2. The general arc creation procedure appears on the left, while the dual graph associated with the graph $\widetilde{G'}$ of Figure 4.1 is drawn on the right with dashed lines. To simplify the drawing, arcs with arrows on both sides represent two oppositely directed arcs.



(the arc of length u_{ij} corresponds to the standard dual arc)

Figure 4.2 Illustration of the construction of $D_{\widetilde{G'}}$ from $\widetilde{G'}$

Remark 4.2.1 *If an arc (i, j) does not separate two faces, it means that (i, j) is a bridge. If we delete (i, j), G' is then non-connected: we call G_1 (resp. G_2) the component with the vertex i (resp. j). Since the arc is from i to j, we have that r is in G_1, otherwise the arc*

(i,j) is useless. If there is at least one terminal t in G_2, then there does not exist a path from r to t after the deletion of (i,j) and hence G' is not k-survivable. Otherwise, the arc (i,j) is useless because G_2 can be disconnected from the graph without impacting the feasibility of the solution. Hence, we consider that there are no bridges in G'.

Zenklusen [68] has shown that there exists a feasible circulation in $\widetilde{G'}$ if and only if there is no negative circuit in $D_{\widetilde{G'}}$. Indeed, there is a correspondence between the circuits in $D_{\widetilde{G'}}$ and the cutsets in $\widetilde{G'}$. In a circuit of $D_{\widetilde{G'}}$, the total length of the standard dual arcs corresponds to the available capacity, while the absolute value of the total length of the other arcs corresponds to the demand. A circuit of negative total length means that the demand cannot be satisfied. For example, consider the circuit (f,d,h,g,f) in the graph $D_{\widetilde{G'}}$ of Figure 4.2: the standard dual arcs (f,d) and (h,g) give a total capacity $u_{ut_1}+u_{t_2t_1}$, while the other arcs (d,h) and (g,f) induce a demand of 1 unit (since $|l_{dh}+l_{gf}| = |-1+0| = 1$), which means that $u_{ut_1}+u_{t_2t_1}$ must be at least equal to 1 to satisfy the demand of terminal t_1.

The above construction shows how to determine if $G'=(V,A')$ has a feasible flow, which is equivalent to deciding whether $\widetilde{G'}$ has a feasible circulation. By definition, G' is k-survivable if and only if there is no subgraph G'' of G', obtained by removing at most k arcs of A', so that $\widetilde{G''}$ has no feasible circulation. Every arc removal is equivalent to paying one unit of a *budget* of k units, and we therefore try to find a subgraph G'' of G' so that $\widetilde{G''}$ has no feasible circulation, without exceeding the budget. Zenklusen [68] (who considers more general budget constraints) shows how this can be done by solving a multi-objective shortest path problem. In our simpler case, we propose to consider a graph $(k+1)D_{\widetilde{G'}}$ that is built as follows. We make $k+1$ copies of $D_{\widetilde{G'}}$, and for every arc (i,j) in A', we consider its standard dual arc $a^d_{ij}=(x,y)$ in $D_{\widetilde{G'}}$, and add links of length 0 from the s-th copy of x to the $(s+1)$-th copy of y $(s=1,\ldots,k)$. Note that if a vertex v is not the tail of any standard dual arc (v,w) in $D_{\widetilde{G'}}$, then each of its neighbors in $D_{\widetilde{G'}}$ is the tail of at least one standard dual arc (since if (v,w) is not a standard arc, then (w,v) is). The resulting graph is denoted by $(k+1)D_{\widetilde{G'}}$. An example for $k=1$ and the graph $D_{\widetilde{G'}}$ of Figure 4.2 is given in Figure 4.3. Since A' contains seven arcs, there are seven arcs linking the first to the second copy of $D_{\widetilde{G'}}$.

We now prove that it is possible to test the k-survivability of G' by solving a series of shortest path problems in $(k+1)D_{\widetilde{G'}}$.

Theorem 4.2.1 *A graph $G'=(V,A')$ is k-survivable if and only if all shortest paths linking a first to a $(k+1)$-th copy of a vertex in $(k+1)D_{\widetilde{G'}}$ have a non-negative length.*

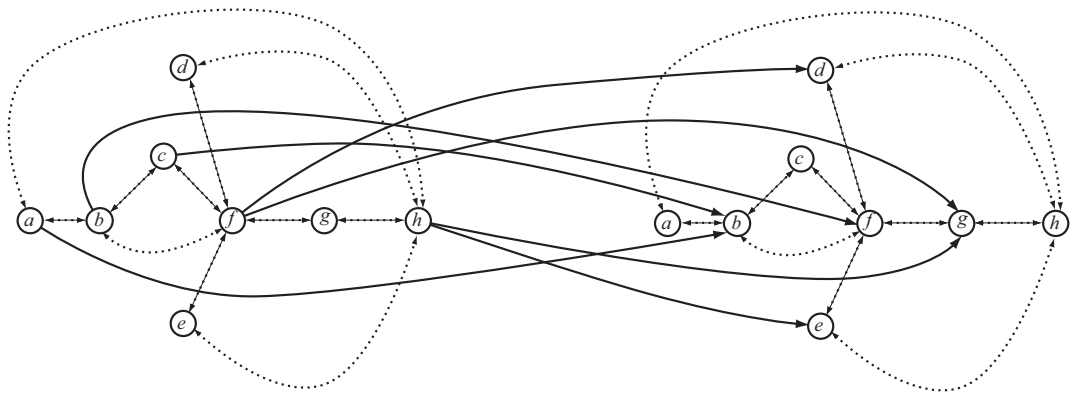


Figure 4.3 Construction of $2D_{\widetilde{G'}}$ for the graph $\widetilde{G'}$ of Figure 4.1

Proof: Assume $G' = (V, A')$ is not k-survivable. As explained previously, this means that there is a circuit C in $D_{\widetilde{G'}}$ and a subset A_C of its arc set such that $|A_C| \leq k$, all arcs in A_C are standard dual arcs, and the total length L of the other arcs on C is strictly negative (because deleting an arc in the primal graph corresponds to contracting the two vertices associated in the dual graph). Let v be a vertex on C which is the tail of at least one standard dual arc (not necessarily on C) in $D_{\widetilde{G'}}$ (note that there always exists such v since we showed that if v is not the tail of any standard dual arcs then each of its neighbors is the tail of at least one standard dual arc). Consider the path P in $(k+1)D_{\widetilde{G'}}$ that links the first to the $(|A_C|+1)$-th copy of v, and uses arcs (x, y) that link consecutive copies of $D_{\widetilde{G'}}$ whenever an arc (x, y) of C belongs to A_C. Clearly, the total length of P is L. Consider any standard dual arc (v, w) in $D_{\widetilde{G'}}$. Note that its opposite arc (w, v) exists and has a non-positive length. We now extend P by adding an arc from the s-th copy of v to the $(s+1)$-th copy of w and an arc for the $(s+1)$-th copy of w to the $(s+1)$-th copy of v, $s = |A_C| + 1, \cdots, k$. Clearly the resulting path P' links the first to the $(k+1)$-th copy of v and has length $L < 0$ in $(k+1)D_{\widetilde{G'}}$.

Suppose now there is a path P of strictly negative length L linking a first to a $(k+1)$-th copy of a vertex v in $(k+1)D_{\widetilde{G'}}$. Let us consider that P contains a proper subpath linking two copies of a vertex w: it covers the case where two arcs in the path linking one copy of $D_{\widetilde{G'}}$ to the next one represent the deletion of the same arc in the input graph.

Suppose that P admits at least one proper subpath linking two copies of a vertex and whose length is strictly negative, and consider a minimal (inclusion-wise) such subpath P' (i.e., no proper subpath of P' links two copies of a vertex). If the total length L' of P' in $(k+1)D_{\widetilde{G'}}$ is strictly negative, then P' corresponds to a circuit C in $D_{\widetilde{G'}}$, and the set A_C of arcs on P' that link two vertices in different copies of $D_{\widetilde{G'}}$ is such that $|A_C| \leq k$, all arcs in A_C are standard dual arcs, and the total length of the other arcs on C is $L' < 0$. Hence G' is not

k-survivable.

So assume all proper subpaths of P that link two copies of a vertex w have a non-negative length. Let $(v = x_1, x_2, \ldots, x_p = v)$ be the sequence of vertices visited by P. If P contains two copies x_i and x_j $(j > i)$ of a vertex with $i \neq 1$ or $j \neq p$, then assume x_i belongs to the s-th copy of $D_{\widetilde{G}'}$, while x_j belongs to the s'-th copy $(s' > s)$. We build a new path P' from P by considering the new sequence $(v = x_1, \ldots, x_i, x_{j+1}, x_{j+2}, \ldots, x_p = v)$ where each $x_q = w$ with $q \geq j+1$ that belongs to the $s_q \geq s'$-th copy of $D_{\widetilde{G}'}$ is replaced by the copy of w lying in the $(s_q + s - s')$-th copy of $D_{\widetilde{G}'}$. Hence, P' is a path linking the first to the $(k + s - s' + 1)$-copy of v, and its length is at most L since we have removed a subpath of P of non-negative length. By repeating this process, we obtain a path P^* linking the first to the k'-th copy of v such that $k' \leq k$, whose length is at most $L < 0$ in $(k+1)D_{\widetilde{G}'}$, and that does not contain any proper subpath linking two copies of a vertex. Hence, P^* corresponds to a circuit C in $D_{\widetilde{G}'}$ and the set A_C of arcs on P^* that link two vertices in different copies of $D_{\widetilde{G}'}$ is such that $|A_C| \leq k$, all arcs in A_C are standard dual arcs, and the total length of the other arcs on C is strictly negative, which means that G' is not k-survivable. $\square$

For illustration of the above theorem, consider the path (f, d, h, g, f) in the graph $2D_{\widetilde{G}'}$ of Figure 4.3, with (h, g) as unique arc in this path linking the first to the second copy of $D_{\widetilde{G}'}$. The length of this path in $2D_{\widetilde{G}'}$ is $u_{ut_1} - 1 + 0 + 0 = u_{ut_1} - 1$, while the corresponding circuit (f, d, h, g, f) in $D_{\widetilde{G}'}$ has length $u_{ut_1} - 1 + u_{t_2t_1} + 0 = u_{ut_1} + u_{t_2t_1} - 1$. Requiring that the total length of the arcs on P must be non-negative is equivalent to impose $u_{ut_1} \geq 1$, which corresponds to the fact that the arc (t_2, t_1) is possibly removed from A', and the flow reaching t_1 can then only come from vertex u.

Theorem 4.2.1 provides an easy way to test k-survivability. Indeed, it is sufficient to determine all shortest paths in $(k+1)D_{\widetilde{G}'}$ linking a first to a $(k+1)$-th copy of a vertex. If one of these paths has a strictly negative length, then the second part of the proof of Theorem 4.2.1 provides a procedure to determine a set A'' of at most k arcs whose removal transforms G' into a graph with no feasible flow. This is summarized in Procedure TestSurvivability which, given an arc set A', either determines a subset A'' of A' such that $|A''| \leq k$ and $G' = (V, A' \setminus A'')$ has no feasible flow, or produces the message "The graph is k-survivable".

4.3 Tabu Search

As shown in [11], CRkECSN is NP-hard, and exact methods can only solve instances of relatively small size. This justifies the use of metaheuristics for larger instances. We propose to

Procedure TestSurvivability(A')

Input : A subgraph $S = (V, A')$ of G;
Output: A subset $A'' \subseteq A'$ with $|A''| \leq k$ such that $G' = (V, A' \setminus A'')$ has no feasible flow, or the message "$G' = (V, A')$ is k-survivable"

```
1  Construct (k+1)D_{G'} and set survivable = true;
2  foreach vertex x in (k+1)D_{G'} do
3      if survivable then
4          Determine a shortest path P linking the first to the (k+1)-th copy of x in
             (k+1)D_{G'};
5          if the total cost of the arcs on P is strictly negative then
6              Apply the procedure used in the second part of the proof of Theorem 4.2.1 to
                 determine a set A'' such that |A''| <= k and G' = (V, A' \ A'') has no feasible
                 flow;
7              Set survivable = false;
8              Go to 12;
9          end
10     end
11 end
12 if survivable then write the message "S = (V, A') is k-survivable";
13 else Return A'';
```

apply to this problem a tabu search, which is one of the most frequently used metaheuristics in combinatorial optimization [34]. The method can be summarized as follows. Let S be the *solution space* (set of feasible solutions) to a combinatorial optimization problem, and let F be a function to be minimized over S. For a solution $s \in S$, let $N(s)$ denote the *neighborhood* of s, which is defined as the set of solutions in S obtained from s by performing a local change, called *move*. A tabu search generates a sequence $s_0, \cdots, s_q$ of solutions in S where s_0 is an initial solution and each s_i $(i = 1, \cdots, q)$ belongs to $N(s_{i-1})$. In order to avoid cycling, the algorithm uses a *tabu list* that contains forbidden moves. Hence, a move m from s_i to s_{i+1} can only be performed if m does not belong to the tabu list, unless $F(s_{i+1})$ is strictly smaller than the value $F(s^*)$ of the best solution encountered so far. A move belonging to the tabu list has a *tabu status* until is is removed from it. As stopping criterion, one may use a fixed amount of CPU time, a fixed number of iterations, a fixed gap to a lower bound value or a fixed number of consecutive iterations without improvement of the value $F(s^*)$. More details are given in [34].

The proposed adaptation of tabu search to CRkECSN can be roughly described as follows. A k-survivable graph is said to be *inclusion-wise minimal* if the removal of any of its arcs makes it non k-survivable. The solution space S explored by the tabu search is the set of

inclusion-wise minimal k-survivable subgraphs G' of the input graph $G = (V, A)$. A move from a solution $G' = (V, A')$ to a solution in its neighborhood is performed as follows. We first remove an arc from A' and put a tabu status on this arc in order to avoid the possibility of cycling, by forbidding to select this arc too early in the current solution again. Since G' is inclusion-wise minimal, this means that the resulting graph is not k-survivable, and we therefore repair it by adding arcs (different from the arcs which have been recently deleted if those are not necessary to a feasible solution) until we obtain a k-survivable subgraph of G. We then remove arcs in order to get a new inclusion-wise minimal k-survivable subgraph. More details about each phase of such a move are given in the next subsections. We will then be ready to give a more precise description of the proposed tabu search.

4.3.1 A repair procedure

In what follows, we denote by c_a the cost of arc a in the input graph G, and by $G' + t$ the graph obtained from a subgraph G' of G by adding a sink t to which all terminals are linked. Let $G' = (V, A')$ be a subgraph of G which has to be repaired in order to become k-survivable. We use the TestSurvivability procedure in order to determine a subset A'' of at most k arcs in A' whose removal transforms G' into a subgraph $G'' = (V, A' \setminus A'')$ of G with no feasible flow. We then determine a maximum flow f in $G'' + t$. This maximum flow has a value $f_{r \to t}$ strictly smaller than $|T|$ (since it is not feasible). In the next step, we build a graph H from $G + t$ as follows:

- all arcs $a \in A' \setminus A''$ for which the flow $f(a)$ is equal to the capacity u_a of a (i.e. saturated arcs) are removed from H; the other arcs of $A' \setminus A''$ are kept in H, but with a zero cost $c_H(a) = 0$. This can be explained by the fact that we do not reroute existing and non-deleted flow, the repair procedure increases the existing flow.

- all arcs $a \in A \setminus A'$ are kept in H. Those with no tabu status have their original cost $c_H(a) = c_a$, while the others have a cost $c_H(a) = c_a + M$, where M is an integer greater than the total cost of the arcs in G.

- all arcs a linking a terminal to the sink t are kept in H and have cost $c_H(a) = 0$.

We next determine a path P of minimum cost linking r to t in H, using cost function c_H. Every arc a on P which does not belong to A' is added to G', which means that by sending one unit of flow on P, the value of the new flow in $G'' + t$ (where $G'' = (V, A' \setminus A'')$ uses the updated arc set A') is exactly one unit larger than the previous one. The idea behind the definition of cost function c_H is to only pay for the addition of arcs not in A': if an arc

$a \in A \setminus A'$ has no tabu status, then the cost of its addition is its original cost c_a, while a penalty M is added to c_a if a has a tabu status (the arc is then only added if it is necessary for the solution to be feasible, otherwise its cost is too high to be selected).

We repeat this process until we get a flow of $|T|$ units in $G'' + t$, which means that, with this new set A' of arcs, the removal of the arcs in A'' does not transform $G' = (V, A')$ into a graph with no feasible flow. There is however possibly another set A''' so that $G'' = (V, A' \setminus A''')$ has no feasible flow. We therefore repeat the complete process until we obtain a k-survivable subgraph of G. The whole repair procedure is summarized in Procedure Repair. Note that, if the original graph G is k-survivable, then Procedure Repair necessarily produces a set A' so that (V, A') is a k-survivable subgraph of G, since the graph $(V, A \setminus A'')$ has a feasible flow for all subsets A'' containing at most k arcs of A.

Procedure Repair(A')

Input : A set $A' \subseteq A$ of arcs such that $G' = (V, A')$ is not k-survivable;

Output: An updated set $A' \subseteq A$ of arcs such that $G' = (V, A')$ is k-survivable;

```
Set ToBeRepaired = true;
Set A'' = TestSurvivability(A') and G'' = (V, A' \ A'');
while ToBeRepaired do
    Determine a maximum flow f in G'' + t and set H equal to G + t;
    foreach a ∈ A' do
        if (a ∈ A'' or f(a) = u_a) then remove a from H;
        else set c_H(a) = 0;
    end
    foreach a ∈ A \ A' do
        if a has a tabu status then set c_H(a) = c_a + M;
        else set c_H(a) = c_a;
    end
    for i = 1 to |T| − f_{r→t} do
        Compute a minimum cost path P (using c_H) linking r to t in H
        foreach a ∈ P do
            if a ∉ A' then
                Add a to A';
                if u_a > 1 then set c_H(a) = 0 and f(a) = 1;
                else remove a from H;
            else
                if f(a) < u_a − 1 then set f(a) = f(a) + 1;
                else remove a from H;
        end
    end
    if the output of TestSurvivability(A') is an arc set A'' then set G'' = (V, A' \ A'');
    else set ToBeRepaired = false;
end
```

4.3.2 Inclusion-wise minimal solutions

Given a k-survivable subgraph $G' = (V, A')$ of G, we now explain how to remove arcs from A' in order to obtain an inclusion-wise minimal solution. Following the notations of Section

2, let $D_{\widetilde{G'}}$ be the graph that contains a negative circuit if and only if G' has no feasible flow. For every arc $(i,j) \in A'$, we consider its standard dual arc $a^d_{ij} = (x,y)$ and determine a shortest path P linking the first copy of y to the $(k+1)$-th copy of x in $(k+1)D_{\widetilde{G'}}$. Let P' be the path in $(k+2)D_{\widetilde{G'}}$ obtained from P by adding the arc linking the $(k+1)$-th copy of x to the $(k+2)$-th copy of y. Clearly, P and P' have the same length. Hence, if P has a negative length, this means that there is a path of negative length in $(k+2)D_{\widetilde{G'}}$ linking the first to the last copy of y, and this path contains an arc linking x to y in different copies of $D_{\widetilde{G'}}$. In other words, the negative length of P' means that there is a graph obtained from G' by removing (i,j) and k additional arcs which does not contains a feasible flow. This means that, if (i,j) is removed from G', then the resulting graph is not k-survivable.

If follows that a procedure that transforms G' into an inclusion-wise minimal k-survivable subgraph of G simply consists in considering all standard dual arcs $a^d_{ij} = (x,y)$ in $D_{\widetilde{G'}}$ and determining a shortest path P linking the first copy of y to the $(k+1)$-th copy of x in $(k+1)D_{\widetilde{G'}}$: if P has a negative length, this means that (i,j) must stay in A' since its removal would make G' not k-survivable; if P has a non-negative length, then (i,j) can be removed from G'.

The procedure that transforms G' into an inclusion-wise minimal solution is called MakeMinimal and is described here below.

Procedure MakeMinimal(A')

Input : A subset $A' \subseteq A$ of arcs so that $G' = (V, A')$ is k-survivable;
Output: An updated subset A' so that $G' = (V, A')$ is k-survivable and inclusion-wise minimal;

```
foreach (i,j) ∈ A' do
    Consider the standard dual arc a^d_ij = (x,y) of (i,j) and compute a shortest path P from
      the first copy of y to the (k+1)-th copy of x in (k+1)D_G'~;
    if P has a non-negative total length then remove (i,j) from A' ;
end
```

Note that there are possibly more than one inclusion-wise minimal subgraphs of G', and the order in which the arcs are considered for possible deletion may therefore have an impact on the resulting graph. We suggest an ordering that gives preference to large costs, but with a random component to bring diversity. More precisely, we suggest to first multiply each cost c_a by a random number ρ_a uniformly chosen in $]0,1]$, which gives a new cost $c'_a = \rho_a c_a$, and

to then order the arcs by non-increasing values of c'_a.

Remark 4.3.1 *In Procedure* MakeMinimal*, at each iteration on an arc $(i,j) \in A'$, if P has a negative length, it means that there is a set of k arcs A_k such that if $A_k \cup \{(i,j)\}$ is removed from G' there does not exist a feasible flow between the root and the terminals. It is useless then to visit in the* foreach *all arcs of $a_k \in A_k$ because they are obviously necessary to a solution. Therefore, the algorithm is modified in this case to not visit them after.*

4.3.3 The proposed tabu search for the CRkECSN

We now describe in more details the proposed tabu search procedure. In what follows, we denote by $c(A) = \sum_{a \in A} c_a$ the total cost of an arc set A.

We first determine whether the input graph $G = (V, A)$ is k-survivable, using the Test-Survivability procedure. If it turns out that $G = (V, A)$ is not k-survivable, then there is obviously no feasible solution to the CRkECSN instance, and we therefore stop the procedure. Otherwise, we start the tabu search. The initial solution $G' = (V, A')$ is generated by applying the MakeMinimal procedure to $A' = A$.

Then, given any solution $G' = (V, A')$ visited by the tabu search, we select a subset $S_{A'}$ of $\lceil \lambda |A'| \rceil$ arcs in A', where λ is a parameter whose value belongs to $]0, 1]$. For every arc a in $S_{A'}$, we create a new set $S^a_{A'}$ by first removing a from A', then applying the Repair procedure, and finally running the MakeMinimal procedure to get a set $S^a_{A'}$ so that $(V, S^a_{A'}))$ is an inclusion-wise minimal k-survivable subgraph of G. In order to avoid the presence of the arc a in $S^a_{A'}$, we add a temporary penalty M to its cost c_a, this penalty being removed when moving to the next arc in $S_{A'}$. Note that it may happen that a necessarily belongs to $S^a_{A'}$. For example, if $G = (V, A)$ contains only two vertices, one being the root r and the other a terminal t_1, and if only two parallel arcs of capacity 1 link r to t_1, then G is 1-survivable since it contains a feasible flow, even if one of its arcs is removed. However, no proper subgraph of G is 1-survivable. Hence, if one of its arc is removed, the Repair procedure has to add it again to recover 1-survivability. This is the reason why we penalize the insertion of a in $S^a_{A'}$ instead of forbidding it. For the same reason, the arcs with a tabu status can be required to repair $A \setminus \{a\}$, and they are therefore also considered for insertion into $S^a_{A'}$, but with a penalty M.

Once all arcs in $S_{A'}$ are treated, the algorithm moves from A' to the set $S^a_{A'}$ with lowest total

cost, and the arc a that was removed from A' to produce $S^a_{A'}$ gets a tabu status for the next τ iterations, where τ is a parameter of the method. The algorithm stops when a stopping criterion is met (which will be a time limit in our case). A detailed description of the whole process appears in Procedure TabuSearchCRkECSN.

In order not to always choose the same subset $S_{A'}$ of arcs while giving preference to arcs with a large cost, we suggest to use the same kind of techniques as the one proposed for ordering the vertices before applying the MakeMinimal procedure. More precisely, we suggest to compute a value $c'_a = \rho_a c_a$ for each $a \in A'$, where ρ_a is a random number uniformly generated in $]0, 1]$ (this is done in the computations in Section 4.4). We then sort the arcs a of A' by non-increasing values of c'_a, and include the $\lceil \lambda |A'| \rceil$ first ones in $S_{A'}$.

Procedure TabuSearchCRkECSN

Input : A graph $G = (V, A)$ with capacity and cost functions on the arc set u and c respectively, four parameters λ, τ, ρ and M, a positive integer k;

Output: A set A^* of arcs so that (V, A^*) is inclusion-wise minimal k-survivable, or the message "$G = (V, A)$ is not k-survivable";

```
if the output of TestSurvivability(A) is an arc set then
    write the message "G = (V, A) is not k-survivable"
end
else
    Set A' = A, apply MakeMinimal(A') and set A* = A';
    while no stopping criterion is met do
        Generate a subset S_A' of ⌈λ|A'|⌉ arcs in A' and set BestCost = +∞ (the arcs in S_A'
          are the ⌈λ|A'|⌉ first arcs a of A' when the arcs of A' are sorted by decreasing value
          of ρ_a c_a);
        foreach a ∈ S_A' do
            Set S^a_A' = A' \ {a} and c_a = c_a + M;
            Apply Repair(S^a_A');
            Apply MakeMinimal(S^a_A');
            if c(S^a_A') < BestCost then set BestCost = c(S^a_A') and a_best = a;
            Set c_a = c_a − M;
        end
        Set A' = S^{a_best}_A' and assign a tabu status to a_best for τ iterations;
        if c(A') < c(A*) then set A* = A';
    end
end
```

4.4 Computational Experiments

Several exact algorithms for CRkECSN based on an integer programming formulation is described in Chapter 3. It applies to all graphs, hence also to non planar ones. We compare it to our TabuSearchCRkECSN procedure. All experiments are performed on a computer with a 2.40GHz Intel(R) Core(TM) i7-5500U CPU and 16Gb of RAM, and the integer programs are solved using CPLEX (v12.2).

Tests are performed on instances with $|V| \in \{20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100\}$

vertices, and with $|T| \in \{\lceil 0.1|V| \rceil, \lceil 0.2|V| \rceil, \lceil 0.3|V| \rceil, \lceil 0.6|V| \rceil, |V|-1\}$ terminals. To create them, we have generated $|V|$ random vertices in the Euclidean plane $[0,1] \times [0,1]$, and the edges are those of a Delauney triangulation. One vertex, chosen at random, is defined as the root r, while $|T|$ other vertices, chosen at random, are considered as terminals. This gives a total of 60 undirected planar graphs. To get oriented graphs, we have replaced every edge linking two vertices i and j by two arcs (i,j) and (j,i).

Since arcs that are close to the root r typically require larger capacities than the other ones, the capacity u_{ij} of an arc (i,j) is chosen at random in $\{\lceil 0.8|T| \rceil, \lceil 0.6|T| \rceil\}$ if there is a path with at most 2 arcs linking r to i, and in $\{\lceil 0.8|T| \rceil, \lceil 0.6|T| \rceil, \lceil 0.4|T| \rceil, \lceil 0.2|T| \rceil\}$ otherwise. According to our experiments, these capacities are high enough to ensure, in most cases, the existence of a 1-survivable subgraph, but are also low enough to make the CRkECSN instances difficult to solve.

For the costs, we have chosen values that depend on the length (in the Euclidean plane) and the capacities of the arcs. More precisely, for two vertices i and j with coordinates (x_i, y_i) and (x_j, y_j), we have set $c_{ij} = \frac{1}{|T|} u_{ij} \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$, which means that the Euclidian distance is multiplied by 0.8, 0.6, 0.4 or 0.2.

Seven of these 60 instances did not pass the test of 1-survivability at the first line of Procedure TabuSearchCRkECSN. These instances therefore do not appear in our experiments. The tabu search was run 10 times on the 53 remaining instances and each execution was stopped after 45 seconds, while we have allocated 3000 seconds to each run of the exact methods described in Chapter 3. Based on preliminary experiments, we have used $\lambda = 0.4$ and $\tau = \left\lceil \sqrt{|A|}/2 \right\rceil$ respectively at lines 7 and 15 of TabuSearchCRkECSN (see Subsection 4.3.3).

Results for $k = 1$ appear in Table 4.1. The $|V|$ and $|T|$ columns indicate respectively the number of vertices and terminals of each instance. Columns C_{opt} and T_{opt} indicate the cost of an optimal solution and the time (in seconds) needed by the exact method to find it, respectively. If no proof of optimality was obtained after 3000 seconds, we indicate the cost of the best feasible solution found as well as, in brackets, the best lower bound on the value of an optimal solution. Column C_b indicates the best cost obtained after 10 runs of the tabu search, while column T_b gives the best time needed to find such a solution, and column n_b shows the number of runs for which a solution of that cost was found. Gray boxes in column

C_b mean that we could reach the proven optimal solution, while gray boxes in column n_b indicate that the 10 runs of our tabu search all ended with the same cost. Column C_a gives the average cost produced by our tabu search, over the 10 runs. While each run was stopped after 45 seconds, column T_a indicates the average time at which the best solution A^* was last improved (line 16 of Procedure TabuSearchCRkECSN). Column I_a indicates the average number of iterations of our tabu search in 45 seconds.

We observe that TabuSearchCRkECSN found the optimal solution for all instances solved to optimality by the exact method. Moreover, these optimal solutions were obtained for each of the 10 runs, except in one case. Indeed, for the instance with $|V| = 45$ vertices and $|T| = 44$ terminals, our tabu search has found the optimal solution in only 3 of the 10 runs, but the average cost $C_a = 28.03$ is very close to the optimal value $C_{opt} = 27.90$ (which corresponds to an increase of around 0.5% of the cost). We also note that, when the exact method has produced a proven optimal solution (i.e., $T_{opt} < 3000$), the average time T_a needed by TabuSearchCRkECSN to obtain the same optimal cost is significantly smaller than T_{opt}. For the instances not solved to optimality (i.e., $T_{opt} = 3000$), the average cost produced by our tabu search is always strictly better than the best cost found by the exact method. We observe that the average number I_a of iterations of the tabu search decreases when the number $|T|$ of terminals or the number $|V|$ of vertices increases. For graphs with 100 vertices, less than 100 iterations could be performed, in average, which indicates that more time should probably be allocated so that the tabu search has more chance to find solutions of good quality.

Similar results appear in Tables 4.2 and 4.3 for $k = 2$ and $k = 3$, respectively. Among the 53 1-survivable instances, only 31 instances passed the 2-survivability test, and 8 of them passed the 3-survivability test of line 1 in Procedure TabuSearchCRkECSN. Again, we observe that our tabu search found the optimal solution for all instances solved to optimality by the exact method, and these optimal solutions were obtained for each of the 10 runs, except in one case: the instance with $|V| = 45$ vertices, $|T| = 14$ terminals and $k = 2$, where our tabu search has found the optimal solution in only 7 of the 10 runs. Again in this case, the average cost $C_a = 25.71$ is very close to the optimal value $C_{opt} = 25.63$ (which corresponds to an increase of around 0.3% of the cost). For these larger values of k, we see that TabuSearchCRkECSN is able, in 45 seconds, to get very big improvements when compared to the best solutions produced by the exact method in almost one hour. For example, for the instance with $|V| = 100$ vertices, $|T| = 20$ terminals and $k = 3$, the exact method stopped with a best upper bound equal to 54.79, while our tabu search has generated a solution of

value $C_b = 35.82$, and the average cost C_a is equal to 36.43.

Up to this point, all instances had an arc (i, j) in A if and only if $(j, i) \in A$, while in practice, it often happens that only one of them exists in the given graph $G = (V, A)$. We have thus decided to also test our tabu search on instances where each edge of the original undirected graphs is oriented in exactly one of the two possible directions. For each of the 53 1-survivable instances ot Table 4.1, we have deleted one of the two arcs $(i, j), (j, i)$ for each pair of adjacent vertices. The chosen directions were made so that the resulting oriented graphs remained 1-survivable. We have then run our algorithm on these new instances. Clearly, the optimal value for each new instance is at least as large as the value of the same instance where all edges were associated with two arcs with opposite directions. Results appear in Tables 4.4, 4.5 and 4.6. While 53 new instances were 1-survivable, only 26 of them were 2-survivable, and 4 were 3-survivable. We can observe that the removal of exactly one of the two arcs (i, j), (j, i) for each pair of adjacent vertices has a big impact on the optimal value. For example, for $|V| = 35$ vertices, $|T| = 10$ terminals, and $k = 3$, the optimal solution in Table 4.3 has value 34.76, while the optimal cost C_{opt} is 55.91 in Table 4.6. When comparing C_{opt} with C_b we again observe that our tabu search was always able to determine an optimal solution when the exact method stopped with a proven optimal one, except in one case: for the instance with $|V| = 90$ vertices, $|T| = 54$ terminals and $k = 2$, the optimal cost C_{opt} is 93.78, while the best solution produced by our tabu search has cost $C_b = 94.05$ (increase of around 0.2% of the cost), and the average cost is $C_a = 94.25$ (increase of around 0.5% of the cost). Note however that both T_a and T_b are larger than 40, which means that our tabu search improved the best solution A^* less than 5 seconds before the process was stopped. For this particular instance, we have determined that 25 additional seconds would have been sufficient to reach the optimal solution. By allocating 120 seconds instead of 45, we have even been able to reach the optimum in 7 of the 10 runs.

For four instances with $k = 1$, four with $k = 2$, and one with $k = 3$, our tabu search had at least one run out of 10 which did not end with a proven optimal solution. But for all these instances, the mean cost C_a is always very close to the optimal cost C_{opt}.

Table 4.1 Results for $k = 1$ where $(i, j) \in A$ if and only if $(j, i) \in A$.

|V|	|T|	C_{opt}	T_{opt}	C_b	T_b	n_b	C_a	T_a	I_a
20	4	9.61	1.0	9.61	0.0	10	9.61	0.1	8613
20	6	14.26	5.6	14.26	0.0	10	14.26	0.1	3666
20	12	18.29	29.9	18.29	0.0	10	18.29	0.7	1912
20	19	21.79	13.1	21.79	0.2	10	21.79	0.9	1867
25	5	12.43	12.1	12.43	0.1	10	12.43	0.2	2776
25	8	14.27	13.9	14.27	0.2	10	14.27	0.4	2482
25	15	17.50	42.8	17.50	0.6	10	17.50	1.5	1126
30	3	8.49	1.7	8.49	0.0	10	8.49	0.1	5653
30	9	12.49	14.1	12.49	0.1	10	12.49	0.9	1603
30	18	16.40	47.7	16.40	0.5	10	16.40	2.3	1067
30	29	20.65	40.2	20.65	0.8	10	20.65	3.3	401
35	4	5.91	1.6	5.91	0.0	10	5.91	0.1	3998
35	7	14.25	127.7	14.25	0.3	10	14.25	1.2	1114
35	10	15.49	44.4	15.49	0.4	10	15.49	1.0	1063
35	34	19.18	16.8	19.18	2.1	10	19.18	3.7	256
40	4	11.43	48.6	11.43	0.1	10	11.43	0.8	2174
40	8	14.44	27.0	14.44	0.8	10	14.44	3.5	805
40	12	19.96	1971.7	19.96	2.3	10	19.96	16.2	501
40	24	17.14	104.0	17.14	0.8	10	17.14	6.3	350
40	39	20.45	55.5	20.45	6.0	10	20.45	16.4	183
45	4	7.90	112.8	7.90	0.2	10	7.90	1.0	1919
45	9	12.72	848.0	12.72	1.4	10	12.72	14.3	739
45	14	16.54	91.6	16.54	1.8	10	16.54	4.1	389
45	27	24.84 (24.34)	3000	24.78	6.5	4	24.83	22.4	219
45	44	27.90	186.5	27.90	11.5	3	28.03	19.9	131
50	5	12.65	1507.1	12.65	0.9	10	12.65	5.8	700
50	10	18.07 (13.73)	3000	17.31	4.8	8	17.31	13.6	329
50	15	13.78	1090.7	13.78	2.2	10	13.78	10.6	419
50	30	17.19	420.5	17.19	3.5	10	17.19	10.7	192
60	6	20.49 (10.45)	3000	16.75	2.1	7	16.81	15.8	452
60	12	20.04 (15.83)	3000	18.20	2.9	10	18.20	15.6	225
60	18	18.70 (15.33)	3000	17.14	12.6	9	17.14	23.1	192
60	36	25.39 (21.58)	3000	24.16	5.7	2	24.21	18.7	103
70	7	12.98 (10.12)	3000	12.35	2.4	9	12.35	9.7	404
70	14	22.09 (9.28)	3000	18.14	25.6	2	18.25	19.3	147
70	21	19.79 (17.02)	3000	19.21	31.1	1	19.34	21.5	108
70	42	26.60 (18.62)	3000	23.34	14.3	3	23.47	28.4	78
70	69	29.99 (21.59)	3000	26.34	40.9	1	26.61	35.2	41
80	8	14.82 (9.04)	3000	12.34	4.2	9	12.34	16.3	208
80	16	23.72 (9.49)	3000	16.83	9.5	6	16.95	24.5	125
80	24	27.70 (11.91)	3000	21.96	32.9	1	22.12	27.8	82
80	48	25.92 (14.77)	3000	21.75	39.3	1	21.88	30.8	59
80	79	32.76 (18.02)	3000	27.50	30.6	1	28.78	39.4	31
90	9	25.24 (9.59)	3000	18.85	40.2	1	19.09	33.5	103
90	18	28.10 (9.84)	3000	18.61	17.1	2	18.77	28.2	99
90	27	33.90 (7.80)	3000	23.14	34.8	1	23.48	34.0	62
90	54	34.49 (12.07)	3000	24.92	41.5	1	25.41	40.8	45
90	89	32.34 (21.02)	3000	29.36	43.0	1	30.25	41.6	36
100	10	26.21 (9.70)	3000	16.45	25.3	1	16.75	26.5	89
100	20	21.34 (11.10)	3000	16.60	15.1	1	16.66	27.0	82
100	30	31.27 (8.76)	3000	19.96	34.3	1	20.40	34.7	44
100	60	33.68 (14.15)	3000	25.03	40.7	1	25.53	40.6	27
100	99	34.04 (12.59)	3000	31.38	41.2	2	32.69	43.6	19

Table 4.2 Results for $k = 2$ where $(i, j) \in A$ if and only if $(j, i) \in A$.

|V|	|T|	C_{opt}	T_{opt}	C_b	T_b	n_b	C_a	T_a	I_a
20	4	15.23	1.0	15.23	0.0	10	15.23	0.1	4929
20	6	20.93	5.6	20.93	0.0	10	20.93	0.1	2069
25	8	22.03	17.6	22.03	0.2	10	22.03	0.9	1309
25	15	26.49	94.6	26.49	0.4	10	26.49	1.1	616
30	9	19.46	17.1	19.46	0.2	10	19.46	1.6	1021
35	10	23.06	38.6	23.06	1.2	10	23.06	2.7	567
40	4	16.59	154.2	16.59	0.2	10	16.59	1.3	914
40	12	29.05	894.4	29.05	2.5	10	29.05	6.8	257
40	39	35.57	13.3	35.57	2.1	10	35.57	15.4	119
45	4	12.69	164.9	12.69	0.8	10	12.69	2.3	703
45	9	19.00	1425.4	19.00	0.5	10	19.00	4.1	327
45	14	25.63	138.2	25.63	5.9	7	25.71	15.0	175
50	5	17.56	991.7	17.56	2.2	10	17.56	5.5	338
50	10	28.57 (19.43)	3000	25.53	6.5	6	25.57	12.2	139
50	15	23.95 (22.27)	3000	23.33	2.0	10	23.33	7.0	177
60	6	25.72 (16.98)	3000	23.25	11.0	3	23.31	25.2	142
60	18	29.29 (21.97)	3000	26.92	17.4	7	26.92	28.1	100
60	36	40.59 (35.31)	3000	39.29	24.6	2	39.55	32.6	52
70	7	21.19 (10.81)	3000	17.59	3.9	9	17.60	23.7	189
70	14	33.25 (16.96)	3000	27.05	9.1	3	27.16	24.3	74
80	8	26.17 (12.56)	3000	18.84	11.1	8	18.86	21.8	89
90	9	41.56 (12.47)	3000	25.10	23.6	7	25.12	30.9	59
90	18	38.52 (11.56)	3000	28.05	29.0	1	28.40	36.1	47
90	27	47.35 (10.42)	3000	34.85	34.5	1	35.38	40.4	38
90	54	51.67 (10.85)	3000	41.16	41.5	1	41.79	41.3	22
90	89	52.89 (29.09)	3000	51.03	42.8	1	52.35	43.2	17
100	10	36.55 (12.06)	3000	23.29	15.1	1	23.64	33.6	47
100	20	39.24 (11.77)	3000	25.37	29.5	1	25.51	30.3	48
100	30	42.82 (11.99)	3000	32.04	39.6	1	32.80	39.7	31
100	60	50.52 (14.65)	3000	40.51	44.3	1	41.82	42.9	22
100	99	53.51 (26.76)	3000	51.02	45	1	52.55	44.6	16

Table 4.3 Results for $k = 3$ where $(i, j) \in A$ if and only if $(j, i) \in A$.

|V|	|T|	C_{opt}	T_{opt}	C_b	T_b	n_b	C_a	T_a	I_a
35	10	34.76	42.4	34.76	1.2	10	34.76	3.2	320
40	4	23.51	356.1	23.51	0.9	10	23.51	3.6	423
45	4	18.64	335.1	18.64	1.0	10	18.64	2.4	335
60	6	35.78 (22.18)	3000	32.83	7.0	9	32.89	19.0	95
70	7	29.54 (15.36)	3000	25.96	14.9	3	26.03	17.1	74
70	14	44.53 (23.60)	3000	37.93	13.8	3	37.99	22.8	53
100	10	42.54 (15.31)	3000	32.12	38.8	2	32.60	40.7	27
100	20	54.79 (15.89)	3000	35.82	36.9	1	36.43	39.8	24

Table 4.4 Results for $k = 1$ where $(i, j) \notin A$ if $(j, i) \in A$.

|V|	|T|	C_{opt}	T_{opt}	C_b	T_b	n_b	C_a	T_a	I_a
20	4	12.63	11.5	12.63	0.0	10	12.63	0.0	4912
20	6	17.48	9.4	17.48	0.1	10	17.48	0.3	2293
20	12	24.94	16.0	24.94	0.0	10	24.94	1.6	1004
20	19	42.57	11.3	42.57	0.0	10	42.57	0.1	963
25	5	14.33	20.8	14.33	0.0	10	14.33	0.1	1748
25	8	20.45	28.2	20.45	0.0	10	20.45	0.5	1713
25	15	24.24	21.9	24.24	0.3	10	24.24	0.7	665
30	3	8.61	2.3	8.61	0.0	10	8.61	0.0	5479
30	9	20.18	77.0	20.18	0.3	10	20.18	0.9	907
30	18	30.40	23.5	30.40	0.1	10	30.40	0.4	520
30	29	41.76	20.0	41.76	0.3	10	41.76	0.8	312
35	4	9.03	10.2	9.03	0.1	10	9.03	0.2	2984
35	7	20.76	32.2	20.76	0.2	10	20.76	0.4	669
35	10	24.68 (22.46)	3000	24.07	0.4	10	24.07	3.7	538
35	34	50.56	57.2	50.56	1.2	3	50.56	1.6	213
40	4	13.28	116.2	13.28	0.1	10	13.28	0.9	1649
40	8	25.23	199.9	25.23	0.3	10	25.23	0.8	635
40	12	26.82	850.1	26.82	1.1	4	26.83	1.3	352
40	24	31.97	120.3	31.97	1.5	10	31.97	2.3	207
40	39	55.09	62.0	55.09	3.0	10	55.09	4.6	131
45	4	9.06	78.0	9.06	0.2	10	9.06	1.2	1036
45	9	18.14	241.3	18.14	0.8	10	18.14	1.7	449
45	14	26.20	440.5	26.20	5.4	10	26.20	10.9	267
45	27	44.85	162.6	44.85	2.3	10	44.85	4.1	191
45	44	59.17	126.8	59.17	5.5	10	59.17	8.6	102
50	5	13.87	373.5	13.87	0.1	10	13.87	1.1	690
50	10	25.40 (21.34)	3000	24.93	1.4	8	24.93	11.8	296
50	15	26.9 (19.37)	3000	25.85	2.3	7	25.96	3.4	238
50	30	37.74	183.3	37.74	2.5	10	37.74	6.2	141
60	6	24.75 (13.12)	3000	21.73	8.4	9	21.74	17.4	286
60	12	29.93 (26.64)	3000	29.41	5.0	2	29.53	9.1	198
60	18	32.33 (17.13)	3000	29.91	8.1	5	29.93	18.2	169
60	36	44.04	2487.4	44.04	14.3	7	44.06	25.6	101
70	7	19.03 (10.85)	3000	16.93	1.9	10	16.93	10.1	239
70	14	35.52 (7.81)	3000	28.71	19.7	3	28.79	29.4	121
70	21	34.28 (24.00)	3000	32.18	9.9	8	32.19	15.8	94
70	42	54.11 (45.45)	3000	52.23	19.6	1	52.45	24.6	63
70	69	64.24	731.2	64.24	35.6	2	64.32	39.1	52
80	8	20.51 (11.69)	3000	17.04	5.0	6	17.06	18.8	177
80	16	33.52 (5.36)	3000	26.20	22.0	1	26.75	16.6	99
80	24	45.90 (30.29)	3000	39.77	16.1	1	40.49	21.2	68
80	48	49.84 (38.38)	3000	48.61	39.6	1	48.77	37.5	46
80	79	65.31 (64.00)	3000	65.76	44.1	1	66.58	42.4	37
90	9	30.81 (8.32)	3000	23.40	9.7	2	23.51	27.8	83
90	18	41.73 (12.94)	3000	32.29	38.1	1	32.67	32.2	63
90	27	44.92 (17.07)	3000	38.07	26.9	1	38.43	34.0	48
90	54	51.14 (32.73)	3000	48.83	41.1	1	49.68	41.3	31
90	89	73.08 (73.08)	3000	73.94	40.7	1	74.92	42.4	27
100	10	24.27 (9.71)	3000	21.84	11.4	3	22.00	24.8	95
100	20	35.93 (8.98)	3000	27.38	34.3	1	27.71	30.9	70
100	30	46.31 (12.97)	3000	36.78	33.7	1	37.55	39.9	44
100	60	57.19 (37.17)	3000	56.38	40.5	1	57.12	42.7	37
100	99	75.05 (70.55)	3000	76.98	45	1	78.03	44.1	24

Table 4.5 Results for $k = 2$ where $(i, j) \notin A$ if $(j, i) \in A$.

|V|	|T|	C_{opt}	T_{opt}	C_b	T_b	n_b	C_a	T_a	I_a
20	4	26.99	15.8	26.99	0.1	10	26.99	1.8	2084
20	6	29.39	5.9	29.39	0.0	10	29.39	0.0	1343
25	8	34.98	12.0	34.98	0.0	10	34.98	0.1	817
25	15	47.06	18.9	47.06	0.0	10	47.06	0.1	425
30	9	35.88	46.5	35.88	0.4	10	35.88	0.7	451
35	10	41.52 (38.61)	3000	41.34	1.0	4	41.34	2.6	292
40	4	21.51	515.6	21.51	0.5	10	21.51	1.2	749
40	12	43.32	75.0	43.32	1.6	4	43.43	5.7	216
45	4	17.40 (17.23)	3000	17.40	0.5	10	17.40	2.2	527
45	9	35.64	281.2	35.64	1.9	10	35.64	3.3	222
45	14	47.78	190.5	47.78	2.8	8	47.78	13.9	179
50	5	24.84	2313.6	24.84	1.0	10	24.84	3.4	325
50	10	43.90 (38.19)	3000	41.97	2.9	9	42.04	8.6	181
60	6	38.68 (17.79)	3000	33.81	12.2	1	33.91	19.2	124
60	18	52.55 (50.97)	3000	52.23	9.0	10	52.23	18.2	99
60	36	85.78	438.2	85.78	10.4	6	85.80	21.9	51
70	7	32.38 (18.46)	3000	29.82	6.8	4	29.93	11.8	119
70	14	56.24 (9.56)	3000	47.52	16.6	2	47.66	27.7	71
80	8	33.42 (19.38)	3000	29.58	7.1	7	29.63	15.2	84
90	18	61.59 (22.79)	3000	51.37	33.9	1	51.83	38.7	45
90	27	71.56 (46.51)	3000	68.61	40.8	1	68.94	41.8	36
90	54	93.78	1983.3	94.05	42.4	2	94.25	41.4	25
100	10	52.01 (12.48)	3000	36.29	34.6	1	36.94	37.3	36
100	20	58.46 (6.43)	3000	44.51	41.6	1	45.06	41.4	28
100	30	77.10 (40.09)	3000	65.17	43.1	1	65.55	43.3	19
100	60	106.62 (105.55)	3000	106.45	44.9	1	107.35	44.3	10

Table 4.6 Results for $k = 3$ where $(i, j) \notin A$ if $(j, i) \in A$.

V	T	C_{opt}	T_{opt}	C_b	T_b	n_b	C_a	T_a	I_a
35	10	55.91	121.8	55.91	18.8	1	56.92	16.5	222
40	4	33.59	988.4	33.59	1.0	9	33.64	11.2	379
45	4	28.54	2165.7	28.54	0.6	10	28.54	1.8	311
60	6	52.69 (32.67)	3000	49.12	13.2	1	49.24	12.8	109

CHAPTER 5 WIND FARM CABLE LAYOUT OPTIMIZATION WITH CONSTRAINTS OF LOAD FLOW AND ROBUSTNESS

5.1 Introduction

5.1.1 Presentation of the problem

The design of wind farm networks involves many challenges in optimization, see Subsection 1.3.1 in Chapter 1. In this chapter, we consider the problem of designing a robust cabling network of an offshore wind-farm, at minimal cost, once the location of the turbines has already been decided. The work presented here was carried out in collaboration with an EDF engineer.

More precisely, given a set of offshore wind turbines producing a known quantity of energy, we look for an optimal network able to route the energy produced by all the wind turbines to the point of common coupling (PCC), called *root node* thereafter in this chapter, that will collect the energy and dispatch it to the grid. One of the main characteristics of our network is that it should be *robust*, i.e. resilient to the failure of one or several cables: hence, we aim to build the cheapest network that will be able to route, for a given number of any possible breakdowns, all the produced energy from the wind turbines to the root node. An important constraint of our problem is that the flow of energy routed in the network must satisfy the *Load Flow equations*. We will explain these constraints in the next section.

We model the problem by using an undirected graph $G = (V \cup \{r\}, E)$, where r is the root node, each vertex in V represents either a wind turbine or a junction node between two or more cables, and E is the set of all possible "edges" on which a cable can be installed. The location of the root, of the wind turbines and of the possible junction nodes are known, so the lengths of the edges of E are given. There are different types of electric cables: the cost and capacity of a cable depend on the type of cable chosen and on the length of the cable, and each type is defined by a cost per meter and a capacity.

The design of cabling networks of wind farms has been recently investigated, see Subsection 1.3.1 in Chapter 1.

In this chapter, we first consider the problem when no breakdown can occur: we present a mathematical model to solve the problem exactly. We give some results obtained on real data. Then we consider the problem in its full generality and present two mathematical formulations for the robust case: the first one, derived from the problem without any breakdowns, is a mixed-integer linear program that considers the case where the number of breakdowns that can occur is 1; the second one is a bi-level mixed integer linear program which is a compact formulation with few variables and constraints. We first explain briefly the *Load Flow* equations and how to take into account these constraints.

5.1.2 The Load Flow constraints

Approximation of the load flow equations

Load flow studies, also known as power flow studies, are power system analysis. We briefly explain them in this section but we refer the reader to [59] for more information about electrical power system planning, especially Appendix A for the load flow problem. We define a bus as a node of an electrical network (for example circuit breakers, transformers, conductors or capacitors); in our application, a bus is either an inter-connection node or a wind turbine or the substation. In this chapter, buses are referred to as nodes, and cables between two buses as lines.

Given the production capacities of the different generators (in our application, the generators will correspond to the wind turbines), the load flow equations allow to calculate the electrical state of the network i.e. to determine the voltages, real and reactive power flows, and currents in a system (at each node or cable) under given load conditions. They are used in planning studies to determine if and when electrical devices will become overloaded. They can be used for example to verify that demands are met without overloading the different network facilities or to ensure that maintenance plans (for example taking off a line for replacement) can proceed without undermining the security of the system.

The load flow problem consists in finding the voltage magnitude and phase angle at each node, and the real and reactive power flowing through each line of the network. We define n as the number of nodes in the network. The formulation of the load flow problem requires to consider four variables at each node $v = 1, .., n$ of the network:

- P_v: the net active power injection in MW (Mega Watts), which is the difference between

active power generation and active power demand at node v

- Q_v: the net reactive power injection in MVAr (Mega Volt Ampere reactive), which is the difference between reactive power generation and reactive power demand at node v
- U_v: the voltage magnitude in pu (per unit) at node v
- θ_v: the voltage angle in radians at node v.

We can apply Kirchhoff's law (see Chapter 2.4 in [3]) to each node, which results in:

$$I = YU \tag{5.1}$$

$$I_v = \frac{P_v - \iota Q_v}{|U_v|} e^{\iota\theta_v}, \ \forall v = 1, .., n \tag{5.2}$$

where

- $\iota^2 = -1$
- I_v is the net injected current at node v and I is the vector $(I_v)_{v=1,..,n}$
- U is the vector of node voltages
- Y is the nodal admittance matrix of the system, which represents the admittance (measure in Siemens of how easily a device will allow a current to flow) at each node.

I, U and Y have complex components, and $U_v = |U_v|e^{\iota\theta_v}$ is the vth element of U. Using (5.1) and (5.2), we have:

$$\frac{P_v - \iota Q_v}{|U_v|} e^{\iota\theta_v} = \sum_{w=1}^{n} Y_{vw} U_w$$

The Y matrix is symmetrical. The diagonal elements Y_{vv} of Y (self admittance of node v) are equal to the sum of admittances of all nodes w connected to v. The off diagonal elements Y_{vw} (mutual admittance) are equal to the negative sum of the admittances between v and w, and we have

$$Y_{vw} = Y_{wv} = |Y_{vw}|e^{\iota\delta_{vw}} = H_{vw} + \iota B_{vw} = |Y_{vw}| \cos\delta_{vw} + \iota |Y_{vw}| \sin\delta_{vw}$$

where H_{vw} and B_{vw} are respectively called conductance and susceptance at $[v, w]$, and where δ_{vw} is the argument of Y_{vw}. Using (5.1) and (5.2), one can derive with some calculations the so-called *Load Flow* equations (see Appendix A in [59]) for each node $v = 1, .., n$:

$$P_v = \sum_{w=1}^{n} |Y_{vw}||U_v||U_w| \cos(\theta_v - \theta_w - \delta_{vw}) \tag{5.3}$$

$$Q_v = \sum_{w=1}^{n} |Y_{vw}||U_v||U_w| \sin(\theta_v - \theta_w - \delta_{vw}) \tag{5.4}$$

To solve these equations, two of the four variables (P, Q, U and θ) must be known in advance at each node. This formulation results in a non-linear system of equations which requires iterative solution methods. However, for such methods, convergence is not guaranteed.

In order to introduce these constraints in the optimization model, we consider the *Direct Current* (DC) estimations of the load flow. If we consider the Direct Current Load Flow, it is customary to make the following assumptions [59]:

Assumption A

Line resistances (inducing active power losses) are negligible i.e. $H \ll B$, and we can assume that $H_{vw} = 0$ for all $[v, w]$ or equivalently that $Y_{vw} = \iota B_{vw}$ and $|Y_{vw}| = B_{vw}$. Notice that, since $Y_{vw} = |Y_{vw}|(\cos(\delta_{vw}) + \iota \sin(\delta_{vw}))$, the previous assumption implies $cos(\delta_{vw}) = 0$ and thus we have $\delta_{vw} \approx \pi/2$.

Assumption B

Voltage angle differences are small i.e. $\sin(\theta_v - \theta_w) \approx \theta_v - \theta_w$ and $\cos(\theta_v - \theta_w) \approx 1$, for all $v, w = 1, .., n$.

Assumption C

Magnitudes of node voltages are equal to 1.0 per unit (flat voltage profile) i.e. $|U_v| = 1$ for all $v = 1, .., n$.

Using Assumptions A and C and 5.3, we have for each $v = 1, .., n$:

$$P_v = \sum_{w=1}^{n} |Y_{vw}||U_v||U_w| \cos(\theta_v - \theta_w - \delta_{vw}) = \sum_{w=1}^{n} B_{vw} \cos(\theta_v - \theta_w - \frac{\pi}{2})$$

Using $\cos(\theta_v - \theta_w - \frac{\pi}{2}) = \sin(\theta_v - \theta_w)$ and Assumption B we have for each node v in the system:

$$P_v = \sum_{w=1}^{n} B_{vw}(\theta_v - \theta_w) \tag{5.5}$$

Integration of the Load Flow equations in our problem

Regarding the design of a wind farm, we consider here a non-directed graph $G = (V, E)$ where V is the set of n nodes of the network (i.e. the substation r, the wind turbines and the interconnection nodes) and E is the set of edges where we can build a cable. In the following, we will refer to the substation r as the root. We are given for each edge $[v, w] \in E$ a susceptance B_{vw}. However, if we do not build a cable on the edge $[v, w]$ in the final network, we can set B_{vw} to 0. We also consider $\vec{G} = (V, \vec{E})$ which is the bi-directed graph associated with G (i.e. for each edge $[v, w] \in E$ in G, there are the arcs $\{(v, w), (w, v)\} \subset \vec{E}$ in $\vec{G}$).

For each node v different from the root node (i.e. $v \in V \setminus \{r\}$), it is known that $P_v \geq 0$: either $P_v > 0$ if v provides some power injection in the network (i.e. v is a wind turbine in our case), or $P_v = 0$ if v is a junction node. If v is a wind turbine, P_v is known and gives the estimation of the production of energy of the wind turbine v. For the root r, the load flow equations imply that $P_r = -\sum_{v \in V \setminus \{r\}} P_v$. For each $(v, w) \in \vec{E}$, we define:

$$\Pi_{vw} := B_{vw}(\theta_v - \theta_w)$$

as the active power flow through $[v, w]$ from v to w, and thus we have $\Pi_{vw} = -\Pi_{wv}$ and $P_v = \sum_{w=1}^{n} \Pi_{vw}$.

Property 5.1.1 *If the load flow equations 5.5 are satisfied at each node of a subgraph $\hat{G} = (V, \hat{E})$ of $G = (V, E)$, with $\hat{E} \subseteq E$ and B_{vw} set to 0 if no cable is built on the edge $[v, w]$ (i.e. $[v, w] \notin \hat{E}$), then there exists a chain in $\hat{G}$ between each node v producing a positive active power flow (i.e. such that $P_v > 0$) and the root.*

Proof: Assume that there exists $v' \in V$ such that $P_{v'} > 0$ and v' is not connected to the

root node r. Let $G(v') = (V', E')$ be an inclusion-wise maximal connected subgraph of G that contains v'. By assumptions $r \notin V'$. Let us consider the sum

$$S = \sum_{v \in V'} P_v$$

Since $P_{v'} > 0$, $P_v \geq 0 \;\; \forall v \in V \setminus \{r\}$, and $r \notin V'$, we have $S > 0$. However, using Equation (5.5), we also have:

$$S = \sum_{v \in V'} P_v = \sum_{v \in V'} \sum_{w \in V'} \Pi_{vw} = 0$$

since $\Pi_{vw} = -\Pi_{wv}$. Hence a contradiction. □

Property 5.1.1 ensures that if Constraints 5.5 are satisfied in a mathematical program, we do not have to add connectivity constraints to ensure that there exists a path between the root and each wind turbine, since load flow equations will not be satisfied otherwise.

The approximation of the load flow analysis will be used to ensure that, for each feasible solution, the power routed through the cables will not exceed its capacity.

5.2 The problem without breakdowns

In this section, we present the problem when no breakdown can occur on the cables. We first give a mixed-integer linear model to solve the problem, then we present a heuristic to get a good feasible solution. In the following, we are given P_v for every node different from the root node. We recall that if $v \in V$ represents a wind turbine, then P_v corresponds to the active power produced and collected through the network by the wind turbine, otherwise v is either a junction node and $P_v = 0$ or the root node r and P_r is not defined (technically it is equal to $-\sum_{v \in V \setminus \{r\}} P_v$).

5.2.1 Mathematical formulation

In practice, the given undirected support network, $G = (V, E)$ is often the union of a partial grid on n nodes, of some diagonal edges and of the root node linked to a subset of vertices of the grid. Let (i, j) be the position of a vertex v in the grid, a diagonal edge incident to v is an edge linking v to another vertex at the position $(i+1, j+1)$, $(i+1, j-1)$, $(i-1, j+1)$ or $(i-1, j-1)$; see Figure 5.1 for an example. The set of wind turbines is denoted by

$T \subset V \setminus \{r\}$. Hence $V \setminus (T \cup \{r\})$ denotes the set of junction nodes. We also consider Q different types of electrical cables numbered by $q \in [1, ..., Q]$; Q is generally a small number (≤ 3). For each $q \in [1, ..., Q]$ and each $[v, w] \in E$, we denote by c^q_{vw}, the cost of installing a cable of type q on $[v, w]$. This cost depends on the type of cable chosen and on the length of $[v, w]$. The capacity of a cable of type q on $[v, w]$ is denoted by u^q_{vw} ($c^q_{vw} = c^q_{wv}$ and $u^q_{vw} = u^q_{wv}$). Hence, we aim to design the cheapest network such that the capacity on each installed cable is greater than the active power flow routed through this cable, i.e. $\Pi_{uv} \leq u^q_{uv}$ and $\Pi_{vu} \leq u^q_{uv}$ for each edge $[u, v]$ where a cable of type q is installed. Notice that the network must verify the load flow equations.

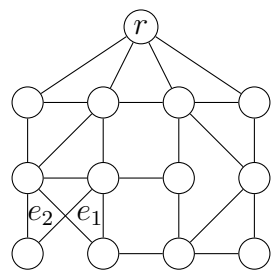


Figure 5.1 A network buildt on a grid 4×3

For technical constraints given by EDF, there is a set $\mathcal{I} \subset E \times E$ of pairs of edges $\{e, e'\}$ such that it is not allowed to install a cable on both e and e'. In practice, the set $\mathcal{I}$ is used to avoid installing cables on two edges that intersect each other, and hence to ensure that the resulting network is planar (on Figure 5.1, we would have for example $\{e_1, e_2\} \in \mathcal{I}$ because e_1 and e_2 are crossing diagonal edges).

5.2.2 Variables and constraints

Recall that the power injection P_v at node $v \in V$ is given for each vertex v. Moreover, B^q_{vw}, corresponding to the susceptance of a cable of type q between v and w, is also given for each $[v, w] \in E$ and each $q \in [1, .., Q]$.

We introduce the following variables:

- For each $q \in [1, ..., Q]$ and for each $e = [v, w] \in E$, let y^q_e be the binary variable such that $y^q_e = 1$ if and only if a cable of type q is installed on $e = [v, w]$. Notice that, in the following, for each $e = [v, w] \in E$, y^q_e can be written indifferently y^q_e, y^q_{vw} or y^q_{wv}.
- For each $v \in V$, let θ_v be the voltage angle at v.

We now introduce the different constraints associated with our problem:

Cable types constraints:

For each $e \in E$, we cannot install more than one type of cable:

$$\sum_{q=1}^{Q} y_e^q \leq 1, \quad \forall e \in E \tag{5.6}$$

Constraints of incompatibility between edges:

The set $\mathcal{I} \subset E \times E$ contains every pair $\{e_1, e_2\}$ such that there cannot be a cable on both edges e_1 and e_2:

$$\sum_{q=1}^{Q} (y_{e_1}^q + y_{e_2}^q) \leq 1, \ \forall (e_1, e_2) \in \mathcal{I} \tag{5.7}$$

Voltage angles constraints:

In the load flow equations, we only consider the differences of angles $\theta_v - \theta_w$ for each pair $(v, w) \in V^2$ with $v \neq w$, i.e. the value of θ_v alone is useless. Using this fact and Assumption B, which states that voltage angles are assumed to be small, we can fix arbitrarily and without loss of generality the value of the angle at the root node r:

$$\theta_r = 0 \tag{5.8}$$

Therefore, since, by Assumption B, for any $v, w \in V$, $\theta_v - \theta_w$ is very small, we have that, for all $v \in V$, $-\varepsilon \leq \theta_v - \theta_r \leq \varepsilon$ for some small $\varepsilon > 0$, which implies:

$$-\varepsilon \ \leq \ \theta_v \ \leq \ \varepsilon \qquad \forall v \in V$$

Differences of voltage angles are assumed to be less than 10^{-1}, hence we have $\varepsilon \approx 10^{-1}$.

Load Flow constraints:

For all $v \in V$, the load flow equations are given by:

$$P_v \quad = \quad \sum_{q=1}^{Q} \sum_{w:[v,w]\in E} B_{vw}^q y_{vw}^q (\theta_v - \theta_w) \tag{5.9}$$

where

$$\sum_{q=1}^{Q} B_{vw}^q y_{vw}^q (\theta_v - \theta_w) \quad = \quad \Pi_{vw}$$

corresponds to the active power sent through $(v, w) \in \vec{E}$. In other words, the load flow constraints ensure that each wind turbine is connected to the root, and allow to calculate the power routed through each cable.

Power line constraints:

For each $e \in E$, the power flow Π_e routed through e must be smaller than the capacity of the cable installed on e:

$$\sum_{q=1}^{Q} B_{vw}^q y_{vw}^q (\theta_v - \theta_w) \quad \leq \quad \sum_{q=1}^{Q} u_{vw}^q y_{vw}^q \qquad \forall (v, w) \in \vec{E} \tag{5.10}$$

One can notice that, if $\sum_{q=1}^{Q} B_{vw}^q y_{vw}^q (\theta_v - \theta_w) < 0$ (resp. > 0), then $\sum_{q=1}^{Q} B_{wv}^q y_{wv}^q (\theta_w - \theta_v) > 0$ (resp. < 0), depending on whether the energy is routed from v to w or from w to v.

5.2.3 Mathematical program

We aim to minimize the total cost of the resulting network, i.e.:

$$\min \sum_{e\in E} \sum_{q=1}^{Q} c_e^q y_e^q$$

We introduce $E_{\mathcal{I}}$ the set of edges e with at least one constraint of incompatibility with another edge, i.e. $e \in E_{\mathcal{I}}$ if there exists at least one edge e' such that $\{e, e'\} \in \mathcal{I}$. The mathematical program to solve can be written as follows:

$$
(\mathbf{LFF}) \left|
\begin{array}{lll}
\min\limits_{y,\theta} & \sum\limits_{e\in E}\sum\limits_{q=1}^{Q} c_e^q y_e^q & \\
\text{s.c.} & \sum\limits_{q=1}^{Q} y_e^q \leq 1, & \forall e \in E \setminus E_{\mathcal{I}} \quad (5.6)\\
& \sum\limits_{q=1}^{Q} (y_{e_1}^q + y_{e_2}^q) \leq 1, & \forall (e_1,e_2) \in \mathcal{I} \quad (5.7)\\
& \theta_r = 0 & (5.8)\\
& \sum\limits_{q=1}^{Q} \sum\limits_{w:[v,w]\in E} B_{vw}^q y_{vw}^q (\theta_v - \theta_w) = P_v, & \forall v \in V \setminus \{r\} \quad (5.9)\\
& \sum\limits_{q=1}^{Q} B_{vw}^q y_{vw}^q (\theta_v - \theta_w) \leq \sum\limits_{q=1}^{Q} u_{vw}^q y_{vw}^q, & \forall (v,w) \in \vec{E} \quad (5.10)\\
& y \in \{0,1\}^{|E|Q}, \quad \theta \in [-\varepsilon, \varepsilon]^{|V|} &
\end{array}
\right.
$$

Using Property 5.1.1, Constraints 5.9 ensure the connectivity between the root and the wind turbines, while Constraints 5.10 ensure that no cable is overloaded.

5.2.4 Linearization of the program

One can notice that the above program have non-linear terms, $y_{vw}^q \theta_v$ and $y_{vw}^q \theta_w$ in Constraints 5.9 and 5.10. Since $-\varepsilon \leq \theta_v \leq \varepsilon \ \forall v \in V$, we have that $0 \leq \theta_v + \varepsilon \leq 2\varepsilon$ for each $v \in V \setminus \{r\}$. We can now propose a well-known solution, to linearize each term $y_{vw}^q(\theta_v + \varepsilon)$ which is the product of a binary variable by a non-negative bounded real variable, see for instance [18]. For all $(v,w) \in \vec{E}$, $q \in [1,...,Q]$, we introduce a non-negative variable ρ_{vw}^q and the following polyhedron:

$$
\mathcal{L}(\theta, y) = \left\{ \rho \in \mathbb{R}^{|\vec{E}|Q} \left|
\begin{array}{lll}
\rho_{vw}^q & \leq & \theta_v + \varepsilon \qquad \forall (v,w) \in \vec{E},\ \forall q \in [1,..,Q]\\
\rho_{vw}^q & \leq & 2\varepsilon y_{vw}^q \qquad \forall (v,w) \in \vec{E},\ \forall q \in [1,..,Q]\\
\rho_{vw}^q & \geq & \theta_v + \varepsilon - 2\varepsilon(1 - y_{vw}^q) \qquad \forall (v,w) \in \vec{E},\ \forall q \in [1,..,Q]
\end{array}
\right. \right\}
$$

We have:

$$
\rho \geq 0,\ \rho \in \mathcal{L}(\theta, y) \quad \Rightarrow \quad \rho_{vw}^q = y_{vw}^q(\theta_v + \varepsilon) \quad \forall (v,w) \in \vec{E},\ \forall q \in [1,..,Q]
$$

Notice that, for each edge $[v,w]$ and each cable type $q \in [1,..,Q]$, we have $\rho_{vw}^q = y_{vw}^q(\theta_v + \varepsilon)$

while $\rho^q_{wv} = y^q_{vw}(\theta_w + \varepsilon)$. Products between θ and y only appear in our problem in the form $y^q_{vw}(\theta_v - \theta_w)$ with $(v, w) \in V^2$. Since we have $y^q_{vw}(\theta_v - \theta_w) = y^q_{vw}\theta_v - y^q_{vw}\theta_w = y^q_{vw}(\theta_v + \varepsilon) - y^q_{vw}(\theta_w + \varepsilon)$, we also have

$$y^q_{vw}(\theta_v - \theta_w) = \rho^q_{vw} - \rho^q_{wv}$$

.

We can then linearize the mathematical program:

$$(\mathbf{LLFF}) \left| \begin{array}{lll} \min\limits_{y,\theta,\rho} & \sum\limits_{e\in E}\sum\limits_{q=1}^{Q} c^q_e y^q_e & \\ \text{s.c.} & \sum\limits_{q=1}^{Q} y^q_e \leq 1, & \forall e \in E \setminus E_{\mathcal{I}} \quad (5.6) \\ & \sum\limits_{q=1}^{Q} (y^q_{e_1} + y^q_{e_2}) \leq 1, & \forall (e_1, e_2) \in \mathcal{I} \quad (5.7) \\ & \theta_r = 0 & (5.8) \\ & \sum\limits_{q=1}^{Q} \sum\limits_{w:[v,w]\in E} B^q_{vw}(\rho^q_{vw} - \rho^q_{wv}) = P_v, & \forall v \in V \setminus \{r\} \quad (5.9) \\ & \sum\limits_{q=1}^{Q} B^q_{vw}(\rho^q_{vw} - \rho^q_{wv}) \leq \sum\limits_{q=1}^{Q} u^q_{vw} y^q_{vw}, & \forall (v, w) \in \vec{E} \quad (5.10) \\ & \rho \in \mathcal{L}(\theta, y) & \\ & y \in \{0,1\}^{|E|Q}, \quad \theta \in [-\varepsilon, \varepsilon]^{|V|}, \quad \rho \geq 0 & \end{array} \right.$$

5.3 Robust approach

In this section we consider that a breakdown may occur on one or several installed cables. We denote by k the maximal number of breakdowns that may occur simultaneously in the network.

5.3.1 The case $k = 1$

Let us denote by $\xi \in E$ the cable where the breakdown occurs. We introduce the following variables:

- For each $q \in [1, ..., Q]$ and for each $e \in E$, let y^q_e be the binary variable such that $y^q_e = 1$ if and only if a cable of type q is installed on e.

- For each $v \in V$ and each $\xi \in E$, let θ_v^ξ be the voltage angle at v in the network where no power can transit through ξ.

We introduce $\vec{\xi}$ for $\xi \in E$ as the set of bi-directed arcs associated with the edge ξ. We propose the following mathematical program to design an optimal network which is robust to one breakdown:

$$
(\kappa) \left|
\begin{array}{lll}
\min\limits_{y,\theta} & \sum\limits_{e \in E} \sum\limits_{q=1}^{Q} c_e^q y_e^q & \\
\text{s.c.} & \sum\limits_{q=1}^{Q} y_e^q \leq 1, & \forall e \in E \setminus E_{\mathcal{I}} \quad (5.11) \\
& \sum\limits_{q=1}^{Q} (y_{e_1}^q + y_{e_2}^q) \leq 1, & \forall (e_1, e_2) \in \mathcal{I} \quad (5.12) \\
& \theta_r^\xi = 0 & \forall \xi \in E \quad (5.13) \\
& \sum\limits_{q=1}^{Q} \sum\limits_{[v,w] \in E \setminus \{\xi\}} B_{vw}^q y_{vw}^q (\theta_v^\xi - \theta_w^\xi) = P_v & \forall v \in V \setminus \{r\},\ \forall \xi \in E \quad (5.14) \\
& \sum\limits_{q=1}^{Q} B_{vw}^q y_{vw}^q (\theta_v^\xi - \theta_w^\xi) \leq \sum\limits_{q=1}^{Q} u_{vw}^q y_{vw}^q & \forall \xi \in E,\ \forall (v,w) \in \vec{E} \setminus \{\vec{\xi}\} \quad (5.15) \\
& y \in \{0,1\}^{|E|Q}, \quad \theta \in [-\varepsilon, \varepsilon]^{|V||E|} &
\end{array}
\right.
$$

In this formulation, θ_v^ξ represents the voltage angle at the node v in the current network when there is a breakdown on the cable built on the edge ξ: we aim to ensure that the load flow equations are satisfied when we cannot route any power flow through this cable. Constraints (5.11) and (5.12) are identical to (5.6) and (5.7), while Constraints (5.13) ensure that θ_r is equal to 0 for each case of breakdown $\xi \in E$. Constraints (5.14) ensure that the load flow is respected (and so each turbine is connected to the root) for any breakdown $\xi \in E$ by not considering the active power on ξ in this case. Indeed, for each scenario of breakdown on a cable built on the edge $\xi \in E$, we do not consider in Constraints (5.14) the edge ξ in the calculation of the sum of energy leaving v, which is equivalent to not considering the cable built on ξ. Constraints (5.15) then ensure that, for any breakdown on $\xi \in E$, the capacities in the resulting network are high enough to support the active power through the cables. The program above hence computes a network of minimal cost which is robust to one breakdown.

Again, we have a product of variables θ and y, which we linearize in a similar way as in the

non-robust model. We introduce $\mathcal{L}^{\xi}(\theta, y)$, which is the linearization of the product $y^q_{uv}(\theta^{\xi}_v+\varepsilon)$, where y^q_{uv} is a binary variable and $(\theta^{\xi}_v + \varepsilon)$ is a non-negative variable, considering the robust case:

$$\mathcal{L}^{\xi}(\theta, y) = \left\{ \rho \in \mathbb{R}^{|\vec{E}||E|Q} \left| \begin{array}{lll} \rho^{q,\xi}_{vw} & \leq & \theta^{\xi}_v + \varepsilon & \forall (v,w) \in \vec{E}, \forall q \in [1,..,Q], \forall \xi \in E \\ \rho^{q,\xi}_{vw} & \leq & 2\varepsilon y^q_{vw} & \forall (v,w) \in \vec{E}, \forall q \in [1,..,Q], \forall \xi \in E \\ \rho^{q,\xi}_{vw} & \geq & \theta^{\xi}_v + \varepsilon - 2\varepsilon(1 - y^q_{vw}) & \forall (v,w) \in \vec{E}, \forall q \in [1,..,Q], \forall \xi \in E \end{array} \right. \right\}$$

The linearized problem becomes:

$$(\mathbf{L}\kappa) \left| \begin{array}{lll} \min\limits_{y,\theta,\rho} & \sum\limits_{e \in E} \sum\limits_{q \in \mathcal{Q}} c^q_e y^q_e & \\ \text{s.c.} & \sum\limits_{q=1}^{Q} y^q_e \leq 1, & \forall e \in E \setminus E_{\mathcal{I}} \quad (5.11) \\ & \sum\limits_{q=1}^{Q} (y^q_{e_1} + y^q_{e_2}) \leq 1, & \forall (e_1, e_2) \in \mathcal{I} \quad (5.12) \\ & \theta^{\xi}_r = 0 & \forall \xi \in E \quad (5.13) \\ & \sum\limits_{q=1}^{Q} \sum\limits_{[v,w] \in E \setminus \{\xi\}} B^q_{vw}(\rho^{q,\xi}_{vw} - \rho^{q,\xi}_{wv}) = P_v, & \forall v \in V \setminus r,\ \xi \in E \quad (5.14) \\ & \sum\limits_{q=1}^{Q} B^q_{vw}(\rho^{q,\xi}_{vw} - \rho^{q,\xi}_{wv}) \leq \sum\limits_{q=1}^{Q} u^q_{vw} y^q_{vw} & \forall (v,w) \in \vec{E}, \xi \in E \quad (5.15) \\ & \rho \in \mathcal{L}^{\xi}(\theta, y) & \\ & y \in \{0,1\}^{|E|Q}, \theta \in [-\varepsilon, \varepsilon]^{|V||E|}, \rho \geq 0 & \end{array} \right.$$

When we allow an arbitrary number of breakdowns, the number of variables and constraints becomes exponential: with k breakdowns, we would have to consider $(|V||E|^k + |E|Q)$ variables before linearization and $(Q|\vec{E}||E|^k + |V||E|^k + |E|Q)$ after. Therefore we end up with an intractable model to solve.

Even for $k = 1$, the number of constraints and columns of $(\mathbf{L}\kappa)$ can be very high depending on the size of the graph. We propose a constraints generation algorithm similar to the one described in Subsection 3.2.2 in Chapter 3 to deal with this case. We initialize $(\mathbf{L}\kappa)_{E_S}$ which corresponds to $(\mathbf{L}\kappa)$ with only a small subset of edges $E_S \subset E$ in Constraints (5.13)-(5.15), i.e.

we define those constraints only for $\xi \in E_S$. We begin to solve the reduced problem $(\mathbf{L}\kappa)_{E_S}$. When a relevant integer feasible solution $(\hat{y}, \hat{\theta}, \hat{\rho})$ is found, we solve a set of sub-problems $(\kappa\text{-sub})_e$ with continuous variables. We define the set of edges selected in the current integer solution $(\hat{y}, \hat{\theta}, \hat{\rho})$

$$\hat{E}_C = \{e \in E \mid \sum_{q=1}^{Q} \hat{y}_e^q = 1\}$$

and $\vec{E}_C$ corresponds to the set of bi-directed arcs associated to $\hat{E}_C$. We define

$$\hat{E}_S = E_S \cap \hat{E}_C$$

corresponding to the intersection between the set of edges which are in the solution $(\hat{y}, \hat{\theta}, \hat{\rho})$ and the set of edges for which the scenario of breakdown has been taken into account at this moment in the algorithm. Finally, we define:

$$\hat{B}_{vw} = \sum_{q=1}^{Q} B_{vw}^q \hat{y}_{vw}^q \qquad \forall (v,w) \in \vec{E}_C$$

$$\hat{u}_{vw} = \sum_{q=1}^{Q} u_{vw}^q \hat{y}_{vw}^q \qquad \forall (v,w) \in \vec{E}_C$$

where $\hat{B}_{vw}$ (respectively $\hat{u}_{vw}$) corresponds to the susceptance (respectively capacity) on the cable built on $[v,w]$. For the integer feasible solution $(\hat{y}, \hat{\theta}, \hat{\rho})$, we introduce the following sub-problem $(\kappa\text{-sub})_e$ for each $e \in \hat{E}_C \setminus \hat{E}_S$:

$$(\kappa\text{-sub})_e \left| \begin{array}{lll} \min\limits_{\theta} & 0 & \\ \text{s.c.} & \theta_r = 0 & \qquad (5.16) \\ & \sum\limits_{w:[v,w]\in E_C\setminus\{e\}} \hat{B}_{vw}(\theta_v - \theta_w) = P_v & \forall v \in V \setminus r \qquad (5.17) \\ & \hat{B}_{vw}(\theta_v - \theta_w) \leq \hat{u}_{vw} & \forall (v,w) \in \vec{E}_C \setminus \{e\} \qquad (5.18) \\ & \theta \in [-\varepsilon, \varepsilon]^{|V|} & \end{array} \right.$$

The formulation $(\kappa\text{-sub})_e$ allows to determine whether if the load-flow constraints are still satisfied if we remove the edge e from the current integer solution, i.e. to ensure that the solution is resilient to a breakdown on e. Indeed, if there is no feasible solution to $(\kappa\text{-sub})_e$, it means that there are no solutions respecting both the load-flow equations and the capacity constraints if we remove the cable on the edge e. We already ensure that the energy is still routed to the sub-station even in the event of a breakdown on any edge in $\hat{E}_S$, and we have

to ensure that this is the case for edges in $\hat{E}_C \setminus \hat{E}_S$. When an integer solution better than the current one is found, we then solve the set of sub-problems $(\kappa\text{-sub})_e$ for $e \in \hat{E}_C \setminus \hat{E}_S$. If one subproblem $(\kappa\text{-sub})_{\bar{e}}$ for a given $\bar{e} \in \hat{E}_C \setminus \hat{E}_S$ does not have any feasible solution, we add $\bar{e}$ to E_S and we add the constraints associated to $(\mathbf{L}\kappa)_{E_S}$. Otherwise, if all subproblems have feasible solutions, the integer solution is feasible for the general problem. Please note that the subproblems $(\kappa\text{-sub})_e$ have only continuous variables.

5.3.2 Bilevel formulation

In this subsection, we give a bilevel formulation for the general case where the number of breakdowns is bounded by k, which is similar to the one proposed in Subsection 3.2.3 of Chapter 3. The bilevel formulation proposed here is also particular in that the second level is a max min problem (it can be seen as a game with a defender and an attacker).

For each $[i,j] \in E$, we introduce a binary variable b_{ij} where $b_{ij} = 1$ if and only if the attacker chooses to delete the arc (i,j). The variables y and θ are defined as in Subsection 5.2.2. We also introduce the variables η_v for each $v \in V \setminus \{r\}$, which correspond to penalty variables used to satisfy the load flow equations. We define the following polyhedron:

$$\mathcal{B}(y) = \{\ b \in \{0,1\}^{|E|} \mid \textstyle\sum_{[i,j]\in E} b_{ij} \leq k\ ;\ b_{ij} \leq \sum_{q\in Q} y_{ij}^q \quad \forall [i,j] \in E\ \}$$

which defines the set of possible scenarios of edge deletions (i.e. the set of constraints of the attacker): at most k edges can be deleted, and those edges must belong to the ones selected by the defender. We also introduce for each $[i,j] \in E$ and $q \in Q$ the notation

$$\beta_{ij}^q = B_{ij}^q (y_{ij}^q - b_{ij})$$

where β_{ij} is equal to B_{ij}^q if a cable of type q is built by the defender on the edge $[i,j]$ and not deleted by the attacker, and to 0 otherwise (since we have $b_{ij} \leq \sum_{q\in Q} y_{ij}^q$).

We then define another polyhedron:

$$\mathcal{X}(y,b) = \left\{ \begin{array}{rcll} \theta_r & = & 0 & \\ \sum_{q=1}^{Q} \sum_{[v,w]\in E} \beta_{vw}^q(\theta_v - \theta_w) + \eta_v = P_v & & \forall v \in V \setminus r & \\ \beta_{vw}^q(\theta_v - \theta_w) & \leq & u_{vw}^q & \forall (v,w) \in \vec{E},\ \forall q \in Q \\ \eta \in \mathbb{R}_+^{|V|-1}, \quad \theta \in [-\varepsilon, \varepsilon]^{|V|} & & & \end{array} \right\} \begin{array}{r} (5.19a) \\ (5.19b) \\ (5.19c) \end{array}$$

This polyhedron $\mathcal{X}(y,b)$ corresponds to the set of load flow and capacity constraints with penalty variables for a given value of y and b, i.e. once the network has been built and the attacker has deleted some edges. We have that β corresponds to the susceptances in the residual network defined by (y,b). Furthermore, at this point we consider that y and b have already been fixed, so β is not a variable. The variables η_v are penalty variables which ensure that the polyhedron is non-empty: the solution where we have $\theta_v = 0$ and $\eta_v = P_v$ for each vertex v is always feasible. The load flow and capacity constraints are satisfied if there exists a feasible solution with $\sum_{v\in V} \eta_v$ is equal to 0.

We propose the following bilevel program:

$$(BIL) \left| \begin{array}{lll} \min\limits_{y\in\{0,1\}^{|A|}} & \sum\limits_{(i,j)\in A} c_{ij} y_{ij} & \\ \text{s.t.} & \sum\limits_{q=1}^{Q} y_e^q \leq 1 & \forall e \in E \setminus E_{\mathcal{I}} \quad (5.20a) \\ & \sum\limits_{q=1}^{Q} (y_{e_1}^q + y_{e_2}^q) \leq 1 & \forall (e_1, e_2) \in \mathcal{I} \quad (5.20b) \\ & f(y) = 0 & (5.20c) \\ & \text{where } f(y) = \max\limits_{b\in\mathcal{B}(y)} \min\limits_{(\theta,\eta)\in\mathcal{X}(y,b)} \sum\limits_{v\in V} \eta_v & (5.20d) \end{array} \right.$$

The defender first builds a network considering the constraints on the number of cables between two nodes and the planarity constraints given by Constraints (5.20a) and (5.20b) respectively. Then, the attacker deletes a set of at most k edges that the defender has built. Then, the defender ensures that the load flow and capacity constraints are still satisfied: he minimizes $\sum\limits_{v\in V} \eta_v$, which is a sum of positive variables. If this sum is equal to 0, it means that

the load flow and capacity constraints are still satisfied; if it is strictly positive, it means that the load flow constraints and the capacity constraints cannot be satisfied at the same time.

An intuition for the solving method is to dualize the min problem in the second level in order to reformulate the max min problem into a max problem and use a constraints generation algorithm similar to the one proposed in Subsection 3.2.3 in Chapter 3.

5.4 Results analysis

In this section, we present the results of the formulations proposed for the design of wind farm cabling networks with load flow constraints. All experiments were performed on a computer with a 2.40GHz Intel(R) Core(TM) i7-5500U CPU and 16GB RAM, using the solver CPLEX version 12.6.1, interfaced with Julia 0.6.0. We used in particular the package *JuMP*, a tool allowing mathematical modeling. For each test, the algorithm has been stopped after 3000 seconds if it has not terminated yet.

We introduce five real or subpart of real data sets $data_{10}$, $data_{23}$, $data_{28}$, $data_{35}$ and $data_{53}$. Each data set $data_{|T|}$ contains a set of the $|T|$ wind turbines, their geographical location as well as the one of the sub-station. The graphs are partial grids with some diagonal edges. For each type of cable, we are given a cost per meter, a capacity and a susceptance.

Table 5.1 gives the results of the tests for the non-robust case (with the formulation (**LLFF**)) and for the robust case with $k = 1$ (with the formulation ($\mathbf{L}\kappa$)). The column I gives the instance on which the formulations are tested. The column $|Q|$ gives the number of types of cables that we consider for the instance. The column gap_f gives the final gap between the best integer solution found and the best lower bound (i.e. 0 if an optimal solution has been found). The column gap_r gives the gap between the best integer solution found and the best lower bound at the root node of the branch-and-cut. Finally, the column $time(s)$ gives the time to find the optimal solution (or 3000 if an optimal solution has not been found in 3000 seconds).

In Table 5.1, the formulation for the non-robust case allows to solve exactly the problem for all instances with a number of wind turbines of at most 35, except for $data_{35}$ with $|Q| = 3$ where it founds an integer solution within a gap of at most 0.04 to the optimal solution. For $data_{53}$, the final gap is 0.02 for $|Q| = 1$, 0.06 for $|Q| = 2$ and 0.14 for $|Q| = 3$. The solving

Table 5.1 Results of the tests for the non-robust case and for the robust case with $k = 1$

		Non-Robust			Robust ($k = 1$)		
I	$\|Q\|$	gap_f	gap_r	$time(s)$	gap_f	gap_r	$time(s)$
$data_{10}$	1	0	0.13	0.12	0	0	2.44
-	2	0	0.1	1.13	0	0.12	2.63
-	3	0	0.11	0.94	0	0.17	10.48
$data_{24}$	1	0	0.07	3.07	0	0.04	84.3
-	2	0	0.17	6.33	0.003	0.34	3000
-	3	0	0.14	8.21	0.04	0.41	3000
$data_{28}$	1	0	0.14	2.75	0	0.05	58.59
-	2	0	0.22	28.1	0	0.17	2349
-	3	0	0.26	77.9	0.02	0.29	3000
$data_{35}$	1	0	0.21	74.9	0	0.18	553
-	2	0	0.17	375	0.1	0.37	3000
-	3	0.04	0.33	3000	-	-	-
$data_{53}$	1	0.02	0.22	224	0.08	0.33	3000
-	2	0.06	0.2	3000	-	-	-
-	3	0.14	0.38	3000	-	-	-

time or the final gap logically increase with the number of types of cables available, but the formulation still manages to find a solution within a reasonable gap from the optimum value.

For the robust case with $k = 1$, the formulation is efficient especially for $|Q| = 1$, where it solves all the instances to the optimum except $data_{53}$, where the gap between the best integer solution and the optimal value is 8%, which sounds reasonable in this case. For $|Q| = 2$, the formulation gets slower and is not efficient on $data_{53}$. However, it allows to solve optimally $data_{10}$ and $data_{28}$ and find an integer solution guaranteed to be within a really small gap of the optimum (0.3 %). For $data_{35}$, the final gap is 10%. For $|Q| = 3$, the formulation is not efficient on $data_{35}$ and $data_{53}$ but find an integer solution which is optimal or at least close to the optimal value (gaps of 4% and 2%).

The robust formulation has a number of variables $|E|$ times bigger than the one of the non-robust one, which logically explains why it is importantly slower. Furthermore, each incrementation of $|Q|$ adds $3|E|$ variables for the non-robust formulation and $2|E||E| + |E|$ variables for the robust formulation with $k = 1$. Logically, the incrementation of $|Q|$ has thus a higher impact on the robust formulation. Furthermore, the case where $|Q| = 1$ corresponds to the case with uniform capacities, which appears to be easier to solve.

CONCLUSION AND RECOMMENDATIONS

This thesis originates from a joint work between the Conservatoire National des Arts et Métiers (CNAM) in Paris, ENSTA ParisTech and Polytechnique Montréal, and addresses the problems of designing networks subject to edge failures after their design. Our main application being the design of wind farm cabling networks, we have been working with an engineer from EDF (largest producer of electricity in France) through a PGMO project (Programme Gaspard Monge pour l'Optimisation, la recherche opérationnelle et leurs interactions avec les sciences des données) of the Fondation Mathématique Jacques Hadamard.

In the following we sum up the main contributions and we outline some future improvements or research directions.

Main contributions

The problems addressed in this thesis take inspiration or revolve around the problem of designing a cabling network of a wind farm with several notions of robustness. Sustainable development being a major goal nowadays, it appears important to focus on several problems concerning renewable energies, including the design of such cabling networks.

In Chapter 2, we focus on the design of arborescence (or rooted tree) networks. In this context, electricity constraints can be formulated as classical flow constraints. We give a complexity theorem and its proof which states that determining whether there is a rooted spanning tree, respecting the capacity constraints (which states for each arc (i, j) that the number of vertices in the subtree rooted at j must not exceed a given value u_{ij}), is $\mathcal{NP}$-Complete, even in the case of uniform capacities. This results extends a result from Papadimitriou [49], where a cost of selection is associated with each arc of the graph, which states that determining whether there exists a rooted tree respecting the capacity constraints and under a given cost of selection is $\mathcal{NP}$-Complete. We then study the design of Steiner arborescences with various notions of robustness: we aim at designing Steiner arborescences while considering the number of terminals disconnected from the root after an arc-deletion in several scenarios. To the best of our knowledge, this problem has not been studied in the literature. We give several formulations for different scenarios and test them on real wind farm data in order to evaluate their impact on the network designed.

In Chapter 3, we introduce a problem called CRkECSN: we aim at designing a minimum-cost network where we are able to route a unit of flow from the root to each terminal respecting the capacities (a limited amount of flow can be routed through each arc) even in the event of the deletion of any subset of k arcs in the network built (k being a given integer). We propose several formulations, including a new bilevel one, where the second level is a $\min - \max$ problem. We also propose an algorithm based on constraints generation and give a method to generate better constraints at each iteration of the algorithm (the enhanced constraint forbids more non-feasible integer solutions than the initial one). We show that the bilevel formulation is a reformulation of another one based on the cut-sets in the graph. We consider the possibility of protecting arcs; those arcs cannot be deleted from the solutions. We give some test results on generated instances and compare the performance of our algorithms.

In Chapter 4, we also study CRkECSN, but in the case of planar graphs. This is motivated by the fact that the wind-farms in offshore environment can often be modeled by grid graphs. The problem of determining whether a graph is k-survivable (i.e. resilient to the deletion of any subset of k arcs) is NP-complete in the general case. However, we derive a theorem which gives properties of the planar dual graph of a k-survivable graph and extend it to solve this problem in polynomial time in planar graphs, by determining a series of shortest paths. Exact methods for CRkECSN can only solve relatively small size instances. For this reason, we also describe a tabu search algorithm that can handle much larger instances, provided they are planar, and we have shown that it typically produces optimal solutions when these are known. Our algorithm has very low computing times, which makes it particularly interesting in practice, for example for the design of survivable wind farm collection networks with hundreds of wind turbines.

In Chapter 5, we study the problem of designing a cabling network for offshore wind farms while respecting the load flow equations. The load flow analysis corresponds to a non-linear system which allows to determine the state of the network (computations of phase voltages, powers and currents at each node or cable of the network), and then ensure that the power routed through each cable is smaller than its maximum capacity. We use a DC current approximation in order to obtain linear equations. Those equations are added to a Mixed-Integer Program (MIP) and we show that, whenever these constraints are satisfied, the load flow equations are satisfied and the final network is connected. We first give a MIP which allows to solve the problem in the non-robust case (where we do not consider the possibility

of breakdowns on the arcs). In a second time, we give a MIP for the case where $k = 1$ breakdown can occur and a constraints generation algorithm to solve it. We test those MIP on real data given for an offshore wind farm. Finally, we give a bilevel formulation designed to solve the problem for general values of k, and give ideas on how to solve it efficiently.

Research directions

The work presented in this thesis has lead to several questions or interesting research paths that would be appealing to investigate in the future.

It appears that the electricity constraints we studied are easier to take into account when we consider arborescence networks. One of the problems that could be worth studying is a more constrained version of CRkECSN, where we want to design a network which is resilient to the deletion of any subset of k arcs and such that the resulting flow must be routed from the root to the terminals through an arborescent sub-network (after these k arcs have been deleted). We have begun to investigate this problem for $k = 1$ on grid graphs with uniform capacity constraints and have found several properties for particular cases. The problem appears to be more difficult to solve than CRkECSN, since the constraint that the flow must be routed through an arborescent sub-network for each scenario of arc deletions seems more difficult to formulate than CRkECSN.

We can also investigate the bilevel program proposed in Chapter 5. This formulation proposes interesting challenges. Bilevel programming formulations are currently widely used in the literature, and our formulation for the second level using slack variables has a number of variables smaller than the one we have tested in this thesis. Once that the *min* problem has been dualized in the second level, it can be interesting to study different ways to solve this formulation.

Another aspect of the real instances that could be interesting to investigate is the stochasticity. For offshore wind farms with equivalent power, the energy produced by each wind turbine is almost always the same, and hence we have considered a fixed energy production at each turbine. However, the actual production depends partly on the demands or, in wind farm networks, on the amount of wind, and it would be an interesting issue to take this uncertainty into account.

REFERENCES

[1] Ajit Agrawal, Philip Klein, and R Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM Journal on Computing*, 24(3): 440–456, 1995.

[2] Ravindra K Ahuja, James B Orlin, and Thomas L Magnanti. Network flows: theory, algorithms, and applications. 1993.

[3] Charles K Alexander and Matthew Sadiku. *Fundamentals of electric circuits*, volume 3. McGraw-Hill New York, 2009.

[4] Esther M Arkin, Nili Guttmann-Beck, and Refael Hassin. The (k, k)-capacitated spanning tree problem. *Discrete Optimization*, 9(4):258–266, 2012.

[5] Nikitas Assimakopoulos. A network interdiction model for hospital infection control. *Computers in biology and medicine*, 17(6):413–422, 1987.

[6] Mourad Baïou and Ali Ridha Mahjoub. Steiner 2-edge connected subgraph polytopes on series-parallel graphs. *SIAM Journal on Discrete Mathematics*, 10(3):505–514, 1997.

[7] Alan Barton. Addressing the problem of finding a single vital edge in a maximum flow graph. 2005.

[8] Cristina Bazgan, Sonia Toubaline, and Daniel Vanderpooten. Efficient determination of the k most vital edges for the minimum spanning tree problem. *Computers & Operations Research*, 39(11):2888–2898, 2012.

[9] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99 (2):351–376, 2004.

[10] Cédric Bentz, Marie-Christine Costa, and Alain Hertz. On the edge capacitated steiner tree problem. *CoRR, abs/1607.07082*, 2016.

[11] Cédric Bentz, Marie-Christine Costa, and Alain Hertz. On the edge capacitated Steiner tree problem. working paper or preprint, February 2017. URL `https://hal.archives-ouvertes.fr/hal-01465403`.

[12] Claude Berge. Graphes et hypergraphes. 1973.

[13] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations research*, 52 (1):35–53, 2004.

[14] Constantin Berzan, Kalyan Veeramachaneni, James McDermott, and Una-May O'Reilly. Algorithms for cable network design on large-scale wind farms. *Massachusetts Institute of Technology*, pages 1–24, 2011.

[15] Hans-Georg Beyer and Bernhard Sendhoff. Robust optimization–a comprehensive survey. *Computer methods in applied mechanics and engineering*, 196(33-34):3190–3218, 2007.

[16] Daniel Bienstock and Gabriella Muratore. Strong inequalities for capacitated survivable network design problems. *Mathematical Programming*, 89(1):127–147, 2000.

[17] M Didi Biha and Ali Ridha Mahjoub. Steiner k-edge connected subgraph polyhedra. *Journal of Combinatorial Optimization*, 4(1):131–144, 2000.

[18] Alain Billionnet. *Optimisation Discrète, de la modélisation à la résolution par des logiciels de programmation mathématique*. January 2007. URL `https://hal.archives-ouvertes.fr/hal-01125300`.

[19] Quentin Botton, Bernard Fortz, Luis Gouveia, and Michael Poss. Benders decomposition for the hop-constrained survivable network design problem. *INFORMS journal on computing*, 25(1):13–26, 2013.

[20] Choaib Bousba and Laurence A Wolsey. Finding minimum cost directed trees with demands and capacities. *Annals of operations research*, 33(4):285–303, 1991.

[21] K Mani Chandy and Tachen Lo. The capacitated minimum spanning tree. *Networks*, 3 (2):173–181, 1973.

[22] Xiuzhen Cheng and Ding-Zhu Du. *Steiner trees in industry*, volume 11. Springer Science & Business Media, 2013.

[23] Geir Dahl and Mechthild Stoer. A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS Journal on Computing*, 10(1):1–11, 1998.

[24] Stuart E Dreyfus and Robert A Wagner. The steiner problem in graphs. *Networks*, 1 (3):195–207, 1971.

[25] Ding-Zhu Du, JM Smith, and J Hyam Rubinstein. *Advances in Steiner trees*, volume 6. Springer Science & Business Media, 2013.

[26] Dingzhu Du and Xiaodong Hu. *Steiner tree problems in computer communication networks*. World Scientific, 2008.

[27] Patrik Fagerfjäll. Optimizing wind farm layout: more bang for the buck using mixed integer linear programming. *Chalmers University of Technology and Gothenburg University*, 2010.

[28] Martina Fischetti and David Pisinger. Optimizing wind farm cable routing considering power losses. *European Journal of Operational Research*, 270(3):917–930, 2018.

[29] Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[30] Greg N Frederickson and Roberto Solis-Oba. Increasing the weight of minimum spanning trees. *Journal of Algorithms*, 33(2):244–266, 1999.

[31] Harold N Gabow, Michel X Goemans, and David P Williamson. An efficient approximation algorithm for the survivable network design problem. *Mathematical programming*, 82(1-2):13–40, 1998.

[32] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of books in the mathematical sciences. W. H. Freeman, 1979.

[33] PM Ghare, Douglas C Montgomery, and WC Turner. Optimal interdiction policy for a flow network. *Naval Research Logistics Quarterly*, 18(1):37–45, 1971.

[34] Fred Glover. Tabu search. *ORSA Journal on computing*, 1(3):190–206, 1989.

[35] Michel X Goemans and Dimitris J Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Mathematical Programming*, 60(1-3):145–166, 1993.

[36] Michel X Goemans and Young-Soo Myung. A catalog of steiner tree formulations. *Networks*, 23(1):19–28, 1993.

[37] Javier Serrano González, Manuel Burgos Payán, Jesús Manuel Riquelme Santos, and Francisco González-Longatt. A review and recent developments in the optimal wind-turbine micro-siting problem. *Renewable and Sustainable Energy Reviews*, 30:133–144, 2014.

[38] M Grotschel, CL Monma, and M Stoer. Design of survivable networks. *Handbooks in Operations Research and Management Science*, 7:617–672, 1995.

[39] Alain Hertz, Odile Marcotte, Asma Mdimagh, Michel Carreau, and Francois Welt. Optimizing the design of a wind farm collection network. *INFOR: Information Systems and Operational Research*, 50(2):95–104, 2012.

[40] FK Hwang, DS Richards, and P Winter. *The Steiner tree problem*. Annals of Discrete Mathematics. Elsevier, Burlington, MA, 1992. URL `http://cds.cern.ch/record/1621452`.

[41] Raja Jothi and Balaji Raghavachari. Approximation algorithms for the capacitated minimum spanning tree problem and its variants in network design. *ACM Transactions on Algorithms (TALG)*, 1(2):265–282, 2005.

[42] Hervé Kerivin and A Ridha Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005.

[43] Panos Kouvelis and Gang Yu. Robust discrete optimization and its applications. 1997.

[44] Eugene L Lawler. *Combinatorial optimization: networks and matroids*. Courier Corporation, 2001.

[45] Weifa Liang. Finding the k most vital edges with respect to minimum spanning trees for fixed k. *Discrete Applied Mathematics*, 113(2-3):319–327, 2001.

[46] Stephen H Lubore, HD Ratliff, and GT Sicilia. Determining the most vital link in a flow network. *Naval Research Logistics Quarterly*, 18(4):497–502, 1971.

[47] Thomas L Magnanti and S Raghavan. Strong formulations for network design problems with connectivity requirements. *Networks*, 45(2):61–79, 2005.

[48] Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.

[49] Christos H Papadimitriou. The complexity of the capacitated tree problem. *Networks*, 8(3):217–230, 1978.

[50] Cynthia A Phillips. The network inhibition problem. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 776–785. ACM, 1993.

[51] AC Pillai, Chick J, Johanning L, Khorasanchi M, and de Laleu V. Offshore wind farm electrical cable layout optimization. *Engineering Optimization*, pages 1–20, 2015. doi: 10.1080/0305215X.2014.992892.

[52] Hans Jürgen Prömel and Angelika Steger. *The Steiner tree problem: a tour through graphs, algorithms, and complexity*. Springer Science & Business Media, 2012.

[53] Wei Qi, Yong Liang, and Zuo-Jun Max Shen. Joint planning of energy storage and transmission for wind energy generation. *Operations Research*, 63(6):1280–1293, 2015.

[54] S Raghavan. *Formulations and algorithms for network design problems with connectivity requirements*. PhD thesis, Massachusetts Institute of Technology, 1994.

[55] Deepak Rajan and Alper Atamtürk. A directed cycle-based column-and-cut generation method for capacitated survivable network design. *Networks*, 43(4):201–211, 2004. ISSN 1097-0037. doi: 10.1002/net.20004. URL `http://dx.doi.org/10.1002/net.20004`.

[56] H Donald Ratliff, G Thomas Sicilia, and SH Lubore. Finding the n most vital links in flow networks. *Management Science*, 21(5):531–539, 1975.

[57] Javier Salmeron, Kevin Wood, and Ross Baldick. Analysis of electric grid security under terrorist threat. *IEEE Transactions on power systems*, 19(2):905–912, 2004.

[58] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

[59] Hossein Seifi and Mohammad Sadegh Sepasian. *Electric Power System Planning: Issues, Algorithms and Solutions*, volume 49. 01 2011. doi: 10.1007/978-3-642-17989-1.

[60] Kenneth Steiglitz, Peter Weiner, and DJ Kleitman. The design of minimum-cost vable networks. *Circuit Theory, IEEE Transactions on*, 16(4):455–460, 1969.

[61] Mechthild Stoer and Geir Dahl. A polyhedral approach to multicommodity survivable network design. *Numerische Mathematik*, 68(1):149–167, 1994.

[62] Eduardo Uchoa, Ricardo Fukasawa, Jens Lysgaard, Artur Pessoa, Marcus Poggi De Aragao, and Diogo Andrade. Robust branch-cut-and-price for the capacitated minimum spanning tree problem over a large extended formulation. *Mathematical Programming*, 112(2):443–472, 2008.

[63] Douglas Brent West. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[64] David P Williamson, Michel X Goemans, Milena Mihail, and Vijay V Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. *Combinatorica*, 15(3):435–454, 1995.

[65] Richard D Wollmer. Some methods for determining the most vital link in a railway network. 1963.

[66] R Kevin Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2):1–18, 1993.

[67] R Kevin Wood. Bilevel network interdiction models: Formulations and solutions. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[68] Rico Zenklusen. Network flow interdiction on planar graphs. *Discrete Applied Mathematics*, 158(13):1441–1455, 2010.