L'optimisation du déploiement des réseaux optiques. Considérations sur l'incertitude de la demande.

Cédric Hervet

▶ **To cite this version:**

Cédric Hervet. L'optimisation du déploiement des réseaux optiques. Considérations sur l'incertitude de la demande.. Optimisation et contrôle [math.OC]. ENSTA ParisTech, 2013. Français. NNT : . pastel-00960733

HAL Id: pastel-00960733
https://pastel.hal.science/pastel-00960733v1

Submitted on 18 Mar 2014

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

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

Thèse de doctorat de l'Ecole Polytechnique

Spécialité

Mathématiques Appliquées

pour l'obtention du titre de docteur de l'Ecole Polytechnique

présentée par

Cédric Hervet

Optimization of optical network deployment. Considerations on demand uncertainty

Soutenue le 18 Décembre 2013 devant le jury composé de

Matthieu Chardy	Encadrant industriel
Pr. Marie-Christine Costa	Directeur de thèse
Dr. Alain Faye	Directeur de thèse
Stanislas Francfort	Encadrant industriel et invité
Dr. Virginie Gabrel	Examinateur
Dr. Olivier Klopfentstein	Invité
Pr. Philippe Mahey	Rapporteur
Pr. Dritan Nace	Rapporteur
Dr. Adam Ouorou	Examinateur
Pr. Frédéric Roupin	Examinateur

Contents

List of Figures

List of Tables

Remerciements

On présente souvent la thèse comme un travail solitaire très entouré. Pour chacune des modestes pierres que j'ai pu apporter à l'édifice pendant 3 ans, toutes existent, de près ou de loin, grâce à d'autres que moi. Je tiens à remercier mes directeurs de thèse, Marie-Christine Costa et Alain Faye, ainsi que mes encadrants industriels Matthieu Chardy et Stanislas Francfort, pour leur confiance, leur gentillesse et leur patience sans laquelle j'aurai eu bien des peines à canaliser mon enthousiasme. Je sais combien leur soutien constant et leur amitié m'a aidé, particulièrement dans les derniers instants, à mener ce travail au bout. Merci encore.

Je remercie chaleureusement Philippe Mahey et Dritan Nace d'avoir accepté d'être les rapporteurs de mon travail de thèse, et Virginie Gabrel, Adam Ouorou, Frédéric Roupin d'en être les examinateurs.

Je remercie également Orange, en particulier Bertrand Decocq et Marie-Françoise Colinas, pour m'avoir donné les moyens d'effectuer ce travail dans des conditions idéales. Je les remercie pour leur bonne humeur et leurs encouragements. Avec eux, je remercie également les collègues (André, Mickaëlle, Joanna, Stéphane, Christian, ...), les thésards (Manu, José, la bande de TRM : le corse, le blog, Amel et tous les autres) et les stagiaires (Dedy, Constant, Joseph, ...) que j'ai pu croiser et dont la liste est bien trop longue pour que je sois exhaustif ici : je sais qu'ils se reconnaîtront.

Je tiens à remercier les membres du laboratoire CEDRIC du CNAM, ainsi que de l'UMA de l'ENSTA, pour m'avoir toujours accueilli avec bonne humeur malgré mon statut de "réfugié industriel". Je pense également aux thésards, notamment Pierre-Louis et Sabine, explorateurs comme moi des recoins méandreux de la robustesse.

On ne cesse jamais d'être thésard, même en dormant. Il aura fallu toute l'amitié de quelques-uns pour m'éviter d'oublier ce qu'est la vraie vie. Je pense bien sûr à Benoît, mon éclectique compère depuis bien longtemps qui, un jour, m'a donné une leçon d'amitié

que je n'oublierai pas. David et sa lowerie qui aura brisé bien des arbalètes. Hugo qui, un jour, refera le monde autrement que depuis son lit. Pierre-Olivier, qui bauguionne à mes cotés depuis chez Berto jusqu'aux Canaries, n'oublie pas dude : "War has changed". Sébastien, dont les manly tears éclipsent encore les "tap-tap" directifs. Bien sûr, Thomas, fouine et plus vieil ami. Lucie, pour nous avoir si stoïquement supporté. Alexandre, papa extatique à la moue ravageuse. Pierrig, à l'intensité d'avant-garde. Et tous les autres, que je n'oublie pas.

Je remercie également les membres de ma famille, notamment mes parents et ma soeur qui, chacun à leur façon, chacun de leur coté, n'ont jamais cessé de me soutenir pendant toutes ses années. J'espère, avec ce travail, les rendre fiers.

Enfin, bien qu'elle l'ignore sans doute, je n'aurai pas pu me donner autant si je n'avais eu besoin de briller aux yeux d'une jolie roturière qui a fait le choix courageux de me suivre, et dont le soutien m'a été indispensable. Merci.

Donnez-moi de la terre... Donnez-moi de la terre à contrer !

Alain Damasio, La Horde du Contrevent

All the stream that's roaring by
Came out of a needle's eye;
Things unborn, things that are gone,
From needle's eye still goad it on.

William Butler Yeats, A Needle's Eye

A Berto,

Ils ont un drapeau noir
En berne sur l'Espoir
Et la mélancolie
Pour traîner dans la vie
Des couteaux pour trancher
Le pain de l'Amitié
Et des armes rouillées
Pour ne pas oublier

Leo Ferré, Les anarchistes

Introduction

With the advent of the Internet, many new services have been developed, thus increasing the bandwidth requirement to a point where the current copper networks tend to reach their natural limitation. On the long run, congestion in the access network (which is the part of the network that is closest to the client) is inevitable. Therefore, for telecommunication operators, the deployment of new network technologies has to be made in order to sustain this increase in bandwidth requirements, directly in the access network, by replacing the former copper network. The related financial investments will be huge, several billions euros will be necessary to deploy the new infrastructure in France alone. This problem is not new for telecommunication operators who engaged this process a few years ago. For Orange, many choices have been made so far, the most important being the replacement of the old fixed access copper network by an optical one. Optical fibers have many advantages: a higher download and upload rate (both being symmetric), a lower signal attenuation per kilometer, etc. Many technologies were available and the choice was made to deploy a specific kind of architecture: the Passive Optical Network, or PON.

PON is a point to multi-point architecture, which means that one fiber leaving the entry point of the access network can serve several clients. This is done by using optical splitters: these are passive equipments (i.e. they do not require to be electrically powered to function) and can split one fiber flow into many, up to their capacity. Moreover, another choice that was made is to deploy fibers from the entry point of the network, called the Optical Line Terminal (or OLT), to its very end, that is the plug in every house and every apartment. This is called a Fiber To The Home deployment, or FTTH.

Such deployments implying tremendous financial investments, and technically complex, it is a key stake for telecommunication operators like Orange to know how to perform these deployments at the lowest cost possible, while ensuring high quality of service and respecting both engineering and regulation rules. Hence, the general problem that this thesis will tackle is the optimization of a passive optical network deployment while taking various aspects of this deployment into account such as the future running costs of the

network, the cabling constraints and, especially, the inherent uncertainty on the future fiber demand.

This problem has already been described for a deterministic context in Mathieu Trampont's Ph.D. thesis in 2010 [50] who introduced it and for which he proposed a modeling based on a description of optical arcs similar to a integer generalized multi-flow problem. This model took several successive optical splitter levels into account and the potential residual capacities of the existing civil engineering trenches. From that modeling, he proposed an optimization of the capital expenditures (CAPEX) of the network, that is roughly the equipment cost, composed of optical fibers and fibers themselves. He proposed a mathematical program to solve this problem, to which he added cuts in order to strengthen his formulation and accelerate his branch-and-bound procedure. An *a posteriori* cabling heuristic was also proposed. This work of Trampont is important since it was the first to describe specifically the optimization of the FTTH PON deployment, which he was able to solve. However, the cabling is not integrated to the global optimization, as well as the fact that the demand actually is uncertain. Finally, the network is only built regarding its deployment cost, and the future maintenance, administration costs of the network were neglected, even though they usually are more important on the long run.

In its very description, that mathematical optimization problem is both a routing and location problem. Thus, it has a theoretical interest, beyond the practical one. Indeed, the work of Ljubic et al. [30], or Gouveia et al. [29] proposed descriptions and algorithms for generic theoretical problems that could be applied to optical network deployment. The strength of their work is to explore the very structure of the problem, of which they identify recurrent structures that they exploit in order to propose column generation algorithms in the integer linear programming paradigm. However, simplifications have been conceded so that fundamental aspects of the problem could be studied, thus making their work difficult to transpose into the field.

Generally speaking, optimization problems do not mix capital expenditures (CAPEX) and operational expenditures (OPEX). Indeed, one is a huge here-and-now investment, while the other is made over time. These costs are not homogeneous, their separate optimization often leads to difficult problems that are very different in their structure (for example a minimal cost network deployment and the optimization of the maintenance rounds). That is why it is often preferred to separate these problems. However, those two questions are highly linked and the topology of the deployed network will have an impact on what one will be able to optimize later, once it is actually deployed. Among

the rare attempts to optimize CAPEX and OPEX costs at the same time, the work of Wang et al. [53] or Jarray et al. [33] can be quoted. The main interest of their approaches is that it enriches the problem's description and allowed to influence the deployment so that future costs are taken into account. But if the philosophy of these approaches is interesting, network deployment problems studied in these works are too far from FTTH PON deployment and their chosen criteria can not be transposed directly.

In network deployment optimization, literature tells us that most of the time, capacity constraints are often a factor of increasing theoretical complexity of a problem. Cabling constraints are, in a way, one step harder than capacity constraints. Hence, many network design problems are solved with relaxed cabling constraints so they will be taken into account by post-processing heuristics. In the precise field of FTTH PON deployment, the work of Kim et al. [34] proposes an optical network deployment modeling in tree-graphs where cabling constraints are taken into account. This enriched model is used to describe various cable cost relaxations in order to solve the problem without cables and approximate the real problem. They also propose a heuristic for solving the whole problem. However, cost relaxation on cables leads to optimality gaps that may be too wide, especially when several billions are at stake. Moreover, even though their heuristic performs quite well most of the time, it can have a pathological behavior on some instances. Even though it interesting, this work does not provide any efficient way to solve the problem without relaxation.

Finally, about the tackling of demand uncertainty for optical network deployment, nothing has been proposed in the literature so far, to the best of our knowledge. Methods do exist, like stochastic programming or robust optimization. Stochastic programming has been introduced by Dantzig in 1955 [23]. It assumes that uncertain data is following well-known probability laws and introduces the concept of recourse variables. In this approach, the decision-maker considers that decisions may be taken in sequence, one after another, and especially some decisions may be taken without knowing the actual scenario that will occur, and some others may be taken right after the revealing of that scenario. The objective of this approach is not to optimize a certain cost, but an expected cost instead. This approach is suited for problems that repeat over time, with few possible scenarios. On the other hand, robust optimization is a complementary approach from stochastic programming in which one frees himself from probability laws hypotheses to only consider variation intervals for uncertain data. Among all possible scenario, a robustness criterion will be optimized, such as the worst case, or the maximum regret. This approach, initiated by Soyster in 1967 [47], recently received a renewed attention from

Ben-Tal et al. [6], or Bertsimas et al. [15]. At the beginning of the years 2000, it knew many developments and was applied to network conceptions problems, like did Atamturk et al. in 2004 [1]. The strong point of this approach is that it allows one to deal with uncertainty with only a minimal number of hypothesis made about it. However, just like stochastic optimization, it often leads to problems that are very complex to solve in practice.

The passive optical network deployment optimization problem is a recent problem for which various solving approaches have been proposed, more or less close to what is expected from a real-life deployment. The enriching of these models passes by taking OPEX costs, cabling constraints and demand uncertainty into account. On these 3 points, taken independently from the problem, the literature furnishes methods and indications on its feasibility. However, there is not study that tackles theses issues in the PON deployment context. This is what motivates the present work. It will consist, from existing models for the problem, in the integration of these new considerations and in the proposal of efficient algorithms to solve them.

After an extensive review of the literature, the first step of my scientific approach was to examine what future network OPEX costs would be relevant to consider in the PON deployment optimization. The objective of this selection being a potential integration to the general problem, and most of all a validation model to assess the effectiveness of that integration on CAPEX costs.

For the second step, I focused on the fiber cabling issue. If a modeling of this problem exists, an efficient solving approach is still lacking and it will be the purpose of this step. The objective was to propose an algorithm that could tackle the PON deployment optimization problem with cabling constraints in a tree-graph.

After that, my efforts were directed towards the demand uncertainty. To that extent, I decided that the modeling of the problem under uncertainty by itself already makes lots of questions arising on how the integration of uncertainty to the problem should be done. This integration was made so that the robust optimization paradigm could be applied to the problem.

Once the modeling of the problem was considered fitting and relevant, I tackled the conception of efficient, and exact solving approaches for this problem. From that study,

I was able to examine in details what makes a solution "robust".

In order to tackle larger problems in size, the next step was to build approached solving methods. I developed several of them, which I compared experimentally in order to decide which one was best suited for the problem.

Chapter 1

Bibliographical study

1.1 The passive optical network design optimisation problem in a capacitated graph

In this section of the bibliographical study, I shall expose the problem as it was introduced in the Ph.D thesis of Mathieu Trampont [50] in 2010, and later published in [21]. I will detail the problem, a model for this problem, and what were the solving approaches considered at the time.

1.1.1 The deployment of a Fiber-To-The-Home network with the Passive Optical Network technology

In the past few years, bandwidth requirements have been increasing a lot, due to the arrival of new internet services such as the Video on Demand or any Peer-to-Peer based services. This led telecommunication operators to work on the deployment of a new technology in their access network which is the part of the network that is closest to the user. The physical support so far for the transportation of electrical information to be used is copper. Originally designed to provide telephonic services, its use has been extended to the Internet, thanks to Asymmetric Digital Subscriber Line (ADSL) technologies for example. However, since these technologies are reaching their inherent limit in capacity, a new physical support for information transportation will be deployed: optical fibers. Since their functioning is of no interest for this study, let us just highlight that it provides a much higher bandwidth capacity with much better properties for a data oriented network, such as upload-download symmetry, a much lower signal attenuation over the distance, and so on.

Thus, replacing the existing copper access network by an optical one is designated under the generic term of a FTTx deployment. Here, the "x" may be replaced by several

terms:

- N for Neighbourhood
- C for Curb
- B for Building
- H for Home

The main difference between those architectures is the length of copper network that will remain for the client, as shown on figure 1.1.

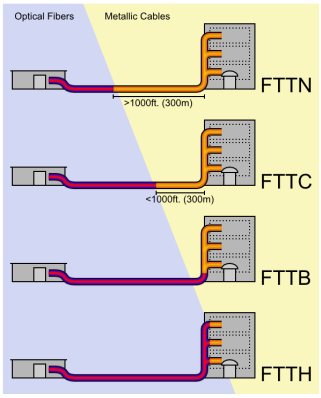

Figure 1.1: FTTx architectures (Source: Wikipedia)

As the access network starts from a point of origin, which is the frontier between the access network and the core network, this distinction on the last letter is here to show what is the range on the network that will be made optical. In our study, we will focus on FTTH network deployment, that is, optical fiber network will be deployed from the entry point (called the Optical Line Terminal, or OLT, for an optical network) of the access network to the very end of it: the user household.

There are several architectures available for the FTTH network deployment, that may be put in 2 categories. First, the point-to-point network architectures. It means that to one fiber coming out from the entry point, one client will be assigned. The other types of architectures is the point-to-multipoint Passive Optical Network (PON) which allows one fiber coming out of the entry point to supply more than one client, thanks to the use of optical splitters which are passive equipments (here, passive must be understood as not needing any power to function) that have the property to split one entering fiber into many, up to a given capacity. In this study, we are focusing on the FTTH PON deployment optimization problem.

Before describing this problem, one needs to understand what a PON architecture is in practice and especially what engineering rules must be respected while deploying such network. As mentioned before, the specific equipment of a PON is the optical splitter equipment. Basically, a PON architecture is composed of several levels of splitters and every client must be connected to the OLT via all splitter levels. The point of using splitters is to reduce the use, at the OLT, of an expensive equipment called PON card. The network operator will have to install one PON card for every pair of fibers coming out of the OLT. Therefore, optical splitters helps reducing the number of PON cards to install, while still supplying the same amount of clients in the area. For a given splitter level, every splitters of that level must have the same capacity, or splitting ratio, that is for every fiber entering a level k splitter, the number of fibers that may go out of that splitter is at most the same for every level k splitter. The figure 1.2 gives an example of a 3 level PON architecture. From the first to the last level of splitters have a capacity equal to 4, 2 and 4. Which means that from 1 fiber coming out of the OLT, 64 clients can potentially be supplied, thus more or less dividing the number of future PON cards to install by 32.

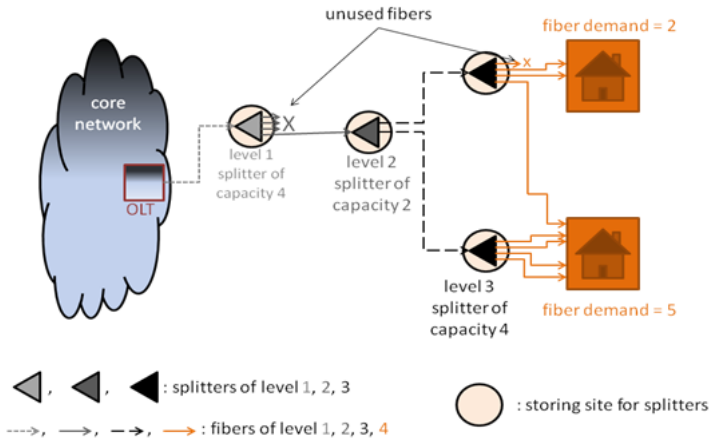

Figure 1.2: An example of a 3 level PON architecture

1.1.2 Description and model of the problem

Focusing on the work of authors of [50, 21], let us describe the problem more into details. Given an area in which one can find a single Optical Line Terminal and a set of potential client households, the problem we aim to solve is to find the FTTH Passive Optical Network that connects every client to the OLT at a minimal cost in terms of installed optical splitters and fibers. In their work, authors considered the hypothesis that the

routing options for fibers are contained in a capacitated non-oriented graph $G = (V, E)$. This means we do not allow new trenches to be dug in order to make the deployment. By convention, we give the index 0 to the OLT node and $|V| = n$. There can be no fiber demand on the OLT node. For the sake of simplicity, we use the notation V^* for $V \backslash \{0\}$ As mentioned before, we need to deploy a k level PON architecture. In practice, there are only 1 to 2 levels of splitters (and, consequently, 1 to 3 levels of fibers) used, because each OLT can only delivers a limited optical budget, that is consumed as the signal travels through fibers, and especially through optical splitters. Let us give the data we need to write the mathematical Modeling of the problem for a 2 level PON architecture:

- c_{ij}^k: cost of a level k fiber routed along the edge (i, j), for all $(i, j) \in E$ and $k = 1, ..., 3$.
- C^k: cost of a level k optical splitter, for $k = 1, 2$.
- $m^k \in \mathbb{N}$: splitting ratio, or capacity, of a level k optical splitter, for $k = 1, 2$ such that $m^k \geq 2$.
- $d_i \in \mathbb{N}$: fiber demand at node i, and more precisely the last level fiber demand at node i, for all $i \in V^*$.
- $b_{ij} \in \mathbb{N}$: capacity of the edge (i, j), for all $(i, j) \in E$, such that $b_{ij} \in \mathbb{N}$.

Integer variables used to describe the possible deployment schemes are the following.

- $\mathbf{z}_i^k$: number of level k optical splitters installed at node i, $\forall i \in V, k = 1, 2$.
- $\mathbf{f}_{ij}^k$: number of level k fibers routed on the edge (i, j) from i to j, $\forall (i, j) \in E, k = 1...3$.
- $\mathbf{u}_i^k$: number of unused level k fibers leaving a splitter of level $k-1$ located at node i, $\forall i \in V, k = 2, 3$

The FTTH Passive Optical Network deployment optimization problem for a 2 level splitter PON architecture can be written as the following integer linear program, denoted by (PON^2):

$$
(PON^2)\begin{cases}
\min\limits_{\mathbf{f},\mathbf{z},\mathbf{u}} & \displaystyle\sum_{i=1}^{n}\sum_{k=1}^{2} C^k\mathbf{z}_i^k + \sum_{(i,j)\in E}\sum_{k=1}^{3} c_{ij}^k\left(\mathbf{f}_{ij}^k + \mathbf{f}_{ji}^k\right) \\
\text{s.t.} & \displaystyle\sum_{j\neq i}\mathbf{f}_{ji}^1 = \mathbf{z}_i^1 + \sum_{j\neq i}\mathbf{f}_{ij}^1,\ \forall i \in V^* & (1.1)\\
& \displaystyle\sum_{j\neq i}\mathbf{f}_{ji}^2 + m^1\mathbf{z}_i^1 = \mathbf{z}_i^2 + \sum_{j\neq i}\mathbf{f}_{ij}^2 + \mathbf{u}_i^2,\ \forall i \in V & (1.2)\\
& \displaystyle\sum_{j\neq i}\mathbf{f}_{ji}^3 + m^2\mathbf{z}_i^2 = d_i + \sum_{j\neq i}\mathbf{f}_{ij}^3 + \mathbf{u}_i^3,\ \forall i \in V & (1.3)\\
& \displaystyle\sum_{k=1}^{3}\left(\mathbf{f}_{ij}^k + \mathbf{f}_{ji}^k\right) \leq b_{ij},\ \forall (i,j) \in E & (1.4)\\
& \mathbf{z}_i^k \in \mathbb{N},\ \ \forall i \in V^*; k = 1,2; \mathbf{z}_0^k = 0, \forall k = 1,2.\\
& \mathbf{u}_i^k \in \mathbb{N},\ \ \forall i \in V^*; k = 2,3; \mathbf{u}_0^k = 0, \forall k = 2,3.\\
& \mathbf{f}_{ij}^k \in \mathbb{N},\ \ \forall (i,j) \in E, k = 1,2,3
\end{cases}
$$

The objective function ensures that we obtain the minimal cost both in terms of installed optical splitters and fibers. Constraints (1.1) ensure that the first level fiber flow is coherent with the number of installed splitters on each nodes: either it goes through, either it is consumed in a level 1 splitter. Constraints (1.2) ensure the same property for the second level of fibers, taking into account that level 2 fibers may be generated by level 1 splitters located on the node and that some of these fibers may be unused. Constraints (1.3) makes sure that the third level of fiber flow is coherent and high enough to supply the demand on every node. Finally, constraints (1.4) makes sure that the whole fiber flow in a given edge does not exceed that edge's capacity.

Integer variables $\mathbf{u}_i^k$ are actually slack variables. Constraints (1.2) and (1.3) may be written as inequalities. Authors wrote the model this way because in practice, solvers were slightly more efficient with the model (PON) given as an entry.

Remark 1.1.1. *For a 1 splitter level PON architecture deployment optimization problem,*

the problem may be written as such:

$$(PON^1)\begin{cases} \min\limits_{\mathbf{f},\mathbf{z},\mathbf{u}} & \sum\limits_{i=1}^{n} C^1\mathbf{z}_i^1 + \sum\limits_{(i,j)\in E}\sum\limits_{k=1}^{2} c_{ij}^k\left(\mathbf{f}_{ij}^k + \mathbf{f}_{ji}^k\right) \\ s.t. & \sum\limits_{j\neq i}\mathbf{f}_{ji}^1 = \mathbf{z}_i^1 + \sum\limits_{j\neq i}\mathbf{f}_{ij}^1,\ \forall i \in V^* & (1.5) \\ & \sum\limits_{j\neq i}\mathbf{f}_{ji}^2 + m^1\mathbf{z}_i^1 = d_i + \sum\limits_{j\neq i}\mathbf{f}_{ij}^2 + \mathbf{u}_i^2,\ \forall i \in V & (1.6) \\ & \sum\limits_{k=1}^{2}\left(\mathbf{f}_{ij}^k + \mathbf{f}_{ji}^k\right) \leq b_{ij},\ \forall (i,j) \in E & (1.7) \\ & \mathbf{z}_i^1 \in \mathbb{N},\ \ \forall i \in V^*; \mathbf{z}_0^1 = 0. \\ & \mathbf{u}_i^2 \in \mathbb{N},\ \ \forall i \in V^*; \mathbf{u}_0^2 = 0. \\ & \mathbf{f}_{ij}^k \in \mathbb{N},\ \ \forall (i,j) \in E, k = 1,2 \end{cases}$$

1.1.3 Remarks, properties and complexity

This section compiles some interesting properties of the problem which can give some insight about it.

Remark 1.1.2. *If fiber costs $c_{ij}^k > 0$, $\forall (i,j) \in E$ and for $k = 1,2,3$, then any optimum solution cannot contain a circuit such that $\mathbf{f}_{ij}^k > 0$ on all arcs of the circuit.*

Proof. If this was not true, one could build a better solution by decreasing by f_{inf} all the $\mathbf{f}_{ij}^k$ along the circuit (f_{inf} being the lowest value of $\mathbf{f}_{ij}^k$ on the circuit). □

Remark 1.1.3. *If optical splitter cost $C^k > 0$ for $k \in \{1,2\}$, then in any optimal solution, we have: $\mathbf{u}_i^{k+1} < m^k,\ \forall i = 1,...,n,\ \forall k = 1,2$.*

Proof. If this was not true, all level k splitters being equivalent, one could consider that one of them produces m^k unused fibers, which trivially leads to non-optimal solutions. □

Remark 1.1.4. *If there are no capacity constraints, integrity constraints for fiber variables $\mathbf{f}_{ij}^k$ can be relaxed in $\mathbf{f}_{ij}^k \geq 0,\ \forall (i,j) \in E,\ \forall k = 1,2,3$.*

Proof. Indeed, for any given integral values of $\mathbf{z}_i^k$ ($i = 1,...,n;\ k = 1,2$), an optimal solution for the routing of fibers can be obtained by solving three independent minimum cost flow problems. Let us consider the constraint matrix of (PON) without capacity constraints (1.4) when variables $\mathbf{z}_i^k$ have fixed integral values; it can be decomposed into three independent flows (totally unimodular) sub-matrices; the right members of the constraints being integral, any extreme point of the polyhedra is integral. This integrity property of variables $\mathbf{f}_{ij}^k$ is no more true either if variables $\mathbf{z}_i^k$ are not integral or if there are capacity constraints. □

Next, authors showed that the problem is NP-hard through a polynomial reduction from the *SET COVER* problem. They work from the single level PON architecture deployment problem which they denote by:

FTTH

INSTANCE: Graph $G = (\{v_0\} \cup V, E)$ where $V = \{v_i,\ i = 1, ..., n\}$ and v_0 as a special node corresponding to the OLT. Demand at node i: d_i, $i = 1, ..., n$. Cost of a fiber of level k on the edge (i, j): $c_{ij}^k,\ i, j = 0, ..., n,\ k = 1, 2$. Cost of an optical splitter: C^1. Splitting ratio of a splitter: m^1. Integer K.

QUESTION: Is there a location of splitters on nodes of V and a routing of fibers following the description given in (PON^1) (remark 1.1.1) such that all demands can be served for a total cost K or less ?

Proposition 1.1.1. *FTTH is NP-complete even if* $d_i = 0$ *or* 1 *for all* i, $b_{ij} \geq \sum_{i \in V} d_i$ *for all* (i, j) *and* $C^1 = 0$.

Proof. The *FTTH* problem is clearly NP. To prove the NP-completeness, we will make the transformation from *SET COVER* which is NP-complete (see [28]):

SET COVER

INSTANCE: Finite set $X = \{x_i; i = 1, ..., n'\}$. Collection of subsets of X: $S = \{S_i; i = 1, ..., n''\}$. Integer K'.

QUESTION: Does S contain a cover for X of size K' or less, i.e., a subcollection $S^* \subseteq S$ with $|S^*| \leq K'$ such that every element of X belongs to at least one member of S^*?

Any set covering problem can be solved using a simplified *FTTH* model. Consider the *FTTH* instance defined on the following graph $G = (V, E)$: $V = \{v_0\} \cup S' \cup X'$ where v_0 represents the OLT, S' contains one node for each subset of S and X' contains one node for each element of X. There is an edge from v_0 to each element of S' and an edge from $S'_j \in S'$ to $x'_j \in X'$ if and only if $x_j \in S_i$. The fiber costs are equal to 1 ($c_{ij}^k = 1$ for all $\{i, j, k\}$). The splitter cost is equal to 0 ($C^1 = 0$). The demand is equal to 0 for any node in S' and to 1 for any node in X'. Let $m^1 = |X'| = n'$. This transformation is made in polynomial time.

Now, with any cover $S^* \subseteq S$ of size $K^* = |S^*|$ we can associate a solution to *FTTH* with value $K^* + n'$ as follows: we assign a splitter to each node in S' associated with a subset of S^* and we route a level 1 fiber from v_0 to each one of these nodes (for a total cost equal to K^*). Then, to each node $x'_i \in X'$, we route a level 2 fiber from one node

in S' with a splitter corresponding to a subset of S^* covering x_i (for a total cost equal to $n_\prime$). Since $m^1 = n'$, each splitter produces a number of level 2 fibers which is large enough to serve all the clients it has to serve (some fibers could be unused) and since all the elements of X are covered in S^*, all nodes in X' received a level 2 fiber in *FTTH*. Finally, since $C^1 = 0$ the cost is equal to $K^* + n'$.

Conversely, let us consider a solution of *FTTH*. Recall that a client must be served by a level 2 fiber. Since $m^1 = |X'| = \sum_i d_i$, w.l.o.g. we can consider solutions with at most one splitter on each node. Moreover, we claim that we can consider only solutions where any client (in X') is served by a splitter installed on an adjacent node of S' and directly linked to v_0 by a level 1 fiber. Indeed, assume that a client in x'_i is served by a splitter in x'_j. If $i \neq j$, let S'_k be the last node in S' along the path followed by the level 2 fiber routed from x'_j to x'_i for a cost of at least 2; we obtain a solution of same cost by assigning a splitter to S'_k if there is none and by routing a level 1 fiber from v_0 to S'_k of S' adjacent to x'_i: we obtain a so good solution by moving the splitter from x'_i to S'_k and linking the splitter to x'_i by a level 2 fiber. Finally, if a level 1 fiber is routed from v_0 to a splitter in $S'_k \in S'$ along a path of length greater than 1, we obtain a better solution by routing directly the level 1 fiber from v_0 to S'_k.

Thus, we can consider a solution of *FTTH* with K^* splitters on nodes of S' linked to v_0 by level 1 fibers (for a cost K^*) and serving demands of nodes in X' by level 2 fibers (for a cost n'). With such a solution (with a total cost $K^* + n'$) we can associate a cover S^* of size K^* containing each subset of S associated with a node with a splitter. Since each client of X' is linked to a splitter in S' each element of X is covered by a subset of S^*. □

1.1.4 Another model for solving the problem with cables in an arborescence

In 2011, Kim, Lee and Han [34] proposed a different model for the optimization of a passive optical network. This model is based on a specific graph topology since they only work in arborescences. However, they tackled the constraint ensuring that fibers are contained in fiber cables of given sizes. Let us review this different formulation, based on paths variables. The notations used in this section will be similar to those used in section 1.1.1.

Let us introduce the model for the PON design problem for a 2 level of splitters architecture in an arborescence. Let $G = (V, A)$ be a tree-graph that contains the set of vertices V and the set of directed arcs A. By convention, we give the index 0 to the root node of the tree, which is the Optical Line Terminal, and we denote by V^* the set of node without the root node. To each leaf node $i \in V^*$ is associated a fiber demand

$d_i \in \mathbb{N}$ (which can be equal to 0 if there is no demand on the node). Let us denote by m^k $(k = 1, 2)$ the capacity of the k-th level of splitters and denote its associated cost by C^k . Fibers can be routed in cables that are listed in a set L, of capacity $b^l \in \mathbb{N}$ for all $l \in L$, and to each arc $(i, j) \in A$ is associated a routing cost the cable of type l, denoted by g^l_{ij}. For any node $i \neq 0$, we denote by S_i the set of node on the path from node i to the root node which is excluded from this set, and we denote by T_i the sub-tree rooted in i.

We define the portion of the demand d_j served by last level splitters placed at node i by the variable $\mathbf{x}_{ij}$. The number of first level splitters placed at node i is denoted by integer variable $\mathbf{z}_i$. The number of second level splitters placed at node j that are served by first level splitters placed at node i is denoted by integer variable $\mathbf{y}_{ij}$. Last, the binary variable $\mathbf{c}^l_{ij}$ is equal to 1 when the cable of type l is installed on the arc (i, j).

The basic model introduced in [34] for the PON deployment of a 2 level architecture optimisation problem is:

$$
(PON^2_{Kim}) \begin{cases}
\min\limits_{\mathbf{c},\mathbf{x},\mathbf{y},\mathbf{z}} & \displaystyle\sum_{l\in L}\sum_{(i,j)\in A} g^l_{ij}\mathbf{c}^l_{ij} + \sum_{i\in V^*} C^1\mathbf{z}_i + \sum_{j\in V^*}\sum_{i\in S_j} C^2\mathbf{y}_{ij} & \\
\text{s.t.} & \displaystyle\sum_{i\in S_j} x_{ij} = 1,\ \forall j \in V^* & (1.8)\\
& \displaystyle\sum_{j\in T_i} \mathbf{y}_{ij} \leq m^1\mathbf{z}_i,\ \forall i \in V^* & (1.9)\\
& \displaystyle\sum_{j\in T_i} d_j\mathbf{x}_{ij} \leq m^2 \sum_{j\in S_i} \mathbf{y}_{ji},\ \forall i \in V^* & (1.10)\\
& \displaystyle\sum_{v\in T_j} \left(\mathbf{z}_v + \sum_{u\in S_i} d_v\mathbf{x}_{uv} + \mathbf{y}_{uv}\right) \leq \sum_{l\in L} b^l\mathbf{c}^l_{ij},\ \forall (i,j) \in A & (1.11)\\
& \displaystyle\sum_{l\in L} \mathbf{c}^l_{ij} \leq 1,\ \forall (i,j) \in A & (1.12)\\
& 0 \leq \mathbf{x}_{ij} \leq 1,\ \forall j \in V^*, \forall i \in S_j & \\
& \mathbf{y}_{ij} \in \mathbb{N},\ \forall j \in V^*, \forall i \in S_j & \\
& \mathbf{z}_i \in \mathbb{N},\ \forall i \in V^* & \\
& \mathbf{c}^l_{ij} \in \{0, 1\},\ \forall (i,j) \in A, \forall l \in L &
\end{cases}
$$

Constraints 1.8 ensure that the demand at node j is satisfied. Constraints 1.9 make sure that the number of first level splitters located at node i has enough capacity to supply last level splitters located in the sub-tree rooted in i. Constraints 1.10 ensure that last level splitters located at node i have enough capacity to supply its portion of the demand located in the sub-tree rooted in i. Constraints 1.11 ensure that the cable installed on the arc (i, j) have enough capacity to contain the whole fiber flow that goes

through the arc. Constraints 1.12 limit the number of cables one can install on an arc to 1.

Remark 1.1.5. *From this model, one can derive a similar problem for a single level architecture. Variables* $\mathbf{y}_{ij}$ *thus have to be removed and the problem* (PON^1_{kim}) *can be formulated as such:*

$$(PON^1_{Kim}) \begin{cases} \min\limits_{\mathbf{c},\mathbf{x},\mathbf{z}} & \sum\limits_{l\in L}\sum\limits_{(i,j)\in A} g^l_{ij}\mathbf{c}^l_{ij} + \sum\limits_{i\in V^*} C^1\mathbf{z}_i \\ s.t. & \sum\limits_{i\in S_j} x_{ij} = 1,\ \forall j \in V^* & (1.13) \\ & \sum\limits_{j\in T_i} d_j\mathbf{x}_{ij} \leq m\mathbf{z}_i,\ \forall i \in V^* & (1.14) \\ & \sum\limits_{v\in T_j}\left(\mathbf{z}_v + \sum\limits_{u\in S_i} d_v\mathbf{x}_{uv}\right) \leq \sum\limits_{l\in L} b^l\mathbf{c}^l_{ij},\ \forall (i,j)\in A & (1.15) \\ & \sum\limits_{l\in L}\mathbf{c}^l_{ij} \leq 1,\ \forall (i,j)\in A & (1.16) \\ & 0 \leq \mathbf{x}_{ij} \leq 1,\ \forall j \in V^*, \forall i \in S_j \\ & \mathbf{z}_i \in \mathbb{N},\ \forall i \in V^* \\ & \mathbf{c}^l_{ij} \in \{0,1\},\ \forall (i,j)\in A, \forall l \in L \end{cases}$$

Authors made the remark that cable variables make the problem quite hard to solve with standard MIP solvers. Therefore, they propose some relaxations of these cabling constraints, mainly by computing an average/minimal/maximal cable cost per fiber in order to have a good approximation/lower bound/upper bound for the problem. They also provide a heuristic for solving the problem on real-size instances.

1.2 The robust approach to optimization under data uncertainty

In this section, I will introduce the main concepts behind optimization under data uncertainty, especially under the scope of robust optimization. Historically, the need for tackling data uncertainty arose from the real-world applications, since in practice, many data are not known precisely. For example, traffic variations in a vehicle routing problem can make travel time uncertain, just like stock exchange variations in portfolio management problems can make costs or gains uncertain too. Therefore, in the optimization context, it was soon admitted that using nominal values, thus denying uncertainty, could lead to bad solutions (in terms of cost), or even to infeasibility. To that extent, the first method used to tackle unreliable data was stochastic optimization. To describe it shortly,

the core aspect of the method is to work under the hypothesis that uncertain data follow well-known probability laws. Therefore, it is possible to describe possible outcomes with a set of scenarios, each one being associated an occurring probability. The goal of stochastic optimization is to maximise the *expected* gain one can expect on that problem. We will not describe stochastic optimization further more, since for this study, the choice was made before starting to focus on robust optimization. Even if this choice can seem arbitrary for our particular problem, it will be justified later and we will show how robust optimization is better suited for the business problem.

1.2.1 The representation of data uncertainty in robust optimization

The first glimpse on robust optimization one can get from the literature comes from Soyster in 1973 [47]. Considering a linear program P:

$$(P)\begin{cases} \max\limits_{x} \sum\limits_{i=1}^{n} c_i x_i \\ \text{s.t.} \sum\limits_{i=1}^{n} a_{ij} x_i \leq b_j, \, \forall j = 1, ..., m \\ x \in X \end{cases}$$

where X is a set of vectors of many variables that can belong to $\mathbb{R}_+^n$ or $\mathbb{N}^n$. The problem raised by Soyster was to consider sets of possible values for each data, instead of a single nominal value. Then, the purpose of the problem was to find a solution that is always feasible. The set representation of data uncertainty will be used as the basis for robust optimization, the robustness coming from the fact that we seek a solution that is always feasible.

Let us consider an uncertain data $\tilde{a}_{ij}$ and the set it will belong to: $[\check{a}_{ij}, \hat{a}_{ij}]$ (this can also be applied to right-hand-side values). Soyster showed that in order to find a solution that is always feasible, one should solve the modified deterministic following problem:

$$(P_{soyster})\begin{cases} \max\limits_{x} \sum\limits_{i=1}^{n} c_i x_i \\ \text{s.t.} \sum\limits_{i=1}^{n} \hat{a}_{ij} x_i \leq \check{b}_j, \, \forall j = 1, ..., m \\ x \in X \end{cases}$$

Indeed, this approach consists in taking the *worst* possible value, that is the more restrictive for the optimization, into account. This approach introduces the set representation of uncertain data. However, despite the fact that this representation frees the decision-maker

from designing complex and often inaccurate probability laws to describe its uncertainty, it was found to be too conservative. This is due to the fact that Soyster's approach makes the solution set much smaller than it was before. Let us take an example with problem $(P1)$:

$$(P1)\begin{cases} \max\limits_{x} \; x_1 + x_2 \\ \text{s.t.} \; x_1 + 2x_2 \leq 8 \\ \qquad 3x_1 + x_2 \leq 10 \\ \qquad x \geq 0 \end{cases}$$

The optimal solution of problem $(P1)$ is $x_1 = 2.4$ and $x_2 = 2.8$ for an optimal gain of 5.2. Let us pose now that every data in constraints of $(P1)$ are uncertain, and that they belong to a set of possible values of size 2, centered in the $(P1)$ value. Therefore, in order to grant feasibility in all cases, one have to solve the problem $(P1_{soyster})$:

$$(P1_{soyster})\begin{cases} \max\limits_{x} \; x_1 + x_2 \\ \text{s.t.} \; 2x_1 + 3x_2 \leq 7 \\ \qquad 4x_1 + 2x_2 \leq 9 \\ \qquad x \geq 0 \end{cases}$$

The optimal solution is such that $x_1 = 1.625$ and $x_2 = 1.25$ for an optimal gain of 2.875, which is almost half the original optimal gain of $(P1)$. This example shows how applying a strict robustness approach can lead to very cost conservative solutions.

1.2.2 The worst case criteria in robust optimization

In 1997, Kouvelis and Yu proposed a unified approach for tackling uncertainty under the name of robust optimization. However, they were not considering a set representation for data uncertainty. Instead, they proposed a scenario based approach, which is close to what is usually done in stochastic optimization. They claimed that decision-makers can easily provide a few representative scenarios regarding the uncertain data, using their field expertise to avoid unrealistic scenarios. But, contrary to the stochastic approach, they make the hypothesis that outcome probability for each scenario may be hard to estimate for a decision-maker, therefore that shall not be expected of him. This is one main difference between stochastic and robust optimization: the existence (or not) of probability assumptions on uncertain data.

From that, they proposed a general concept for robust optimization, stating that a robust solution has, whatever the outcome is, the following properties:

- the solution is always feasible
- the solution cost is "never too high"

If the first item is easy to understand and to formalize, there are many ways of defining "never too high". Formally, one can define any optimization problem (P_{opt}) with x being the decision vector, X its feasible set and f the cost function:

$$(P_{opt}) \begin{cases} \min\limits_{x} & f(x) \\ \text{s.t.} & x \in X \end{cases}$$

Thus, let us consider a set S of possible scenarios for uncertain data. Data can be uncertain in the cost function f, or in the feasibility set X (or both). Therefore, to each scenario s of S, let us denote by f_s and X_s the corresponding cost function and feasibility set. Ensuring a solution x to be feasible in all scenarios can be reduced to imposing that

$$x \in \cap_{s \in S} X_s$$

At this point, their approach of robust feasibility was the same as Soyster's (see the last section 1.2.1). For the cost function, they proposed 3 robustness criteria.

The first one is called the worst case criteria. It means that whatever happens, the worst scenario for the chosen configuration will have the smallest possible cost (w.r.t. a minimization problem). In our previously introduced formalism, this would mean we aim to solve:

$$(P_{wc}) \begin{cases} \min\limits_{x} \max\limits_{s} & f_s(x) \\ \text{s.t.} & x \in \cap_{s \in S} X_s \end{cases}$$

The two other criteria are very similar, and are based on the notion of "regret". That is, contrary to the worst case criterion where one ensures that the solution will not have a higher cost than the worst-case one, the worst regret criterion ensures that the solution cost variation compared to the cost one would have obtained if he knew what the actual scenario would be, won't exceed the worst-regret value. Formally, let us denote by x_s^* the optimal solution of problem (P_{opt}^s) in the case of scenario s. It gives the absolute worst regret criterion:

$$(P_{awr}) \begin{cases} \min\limits_{x} \max\limits_{s} & f_s(x) - f_s(x_s^*) \\ \text{s.t.} & x \in \cap_{s \in S} X_s \end{cases}$$

Which is similar to the proportional worst regret criterion:

$$(P_{pwr}) \begin{cases} \min\limits_{x} \max\limits_{s} & \dfrac{f_s(x) - f_s(x_s^*)}{f_s(x_s^*)} \\ \text{s.t.} & x \in \cap_{s \in S} X_s \end{cases}$$

Of course, regret-based robustness criteria are more computationally demanding, since they require the pre-computing of optimal solutions for every scenario s. However, one can easily see that the average performance of these criteria on the solution cost will be better than with the worst case one. However, that computational barrier will make the worst-case criterion the most used in the literature that follows.

1.2.3 Uncertainty sets for the robust approach

After Kouvelis and Yu [35], the scenario based approach was not extensively used in the literature. Since it requires a good knowledge of the uncertainty in order to define the set of explicit scenarios, the interval approach from Soyster [47] was preferred to model uncertain data. However, the over conservative aspect of this approach led to the design of new uncertainty sets, all based on the assumption that the very worst case (considered in Soyster's approach) has so little chance to happen and is so costly to hedge against that it may be better to just ignore it and restrain the set of possible outcomes.

In terms of Modeling, Ben-Tal and Nemirovski [7] or Bertsimas and Sim [15] proposed a new approach. Given an uncertain data denoted by $\tilde{d}$ belonging to the set of uncertain data D, and denoting by $\bar{d}$ its nominal value, $\hat{d} > 0$ the maximum variation around $\bar{d}$, one can define its uncertainty range as follows:

$$\tilde{d} \in \left[\bar{d} - \hat{d}, \bar{d} + \hat{d}\right]$$

or, by the mean of a random variable $\Delta_d \in [-1, 1]$:

$$\tilde{d} = \bar{d} + \hat{d}\Delta_d$$

Based on the fact that, in practice, the worst configuration (that is either $\tilde{d} = \bar{d} + \hat{d}$ or $\tilde{d} = \bar{d} - \hat{d}$ depending on the problem's optimizing direction and variable signs) is highly conservative and very unlikely to occur, theses authors proposed a way to restrict uncertain data global variations. Babonneau and Vial [2] proposed a comprehensive way to present these approaches.

Ellipsoidal uncertainty sets

Introduced by Ben-Tal et al. [7], it is presented by authors as a natural, intuitive way of limiting uncertainty, that the normalized (that is, considering Δ_d variables) distance from the nominal point $\bar{d}$ should not exceed a given parameter Ω. The uncertainty set $\mathcal{D}_\Omega$ is:

$$\mathcal{D}_\Omega = \left\{ \Delta \in [-1, 1]^{|D|} \,\middle|\, \sqrt{\sum_{d \in D} |\Delta_d|^2} \leq \Omega \right\}$$

This set is defined with only one parameter, Ω, that is very useful when little information is available to the decision-maker. However, this uncertainty set did not become popular in the literature because of the relative difficulty to take it into account in solving approaches. Indeed, as we will see later, its formalism is not suited for linear programming approaches (even though semi-definite programming can be used to solve it), the Bertsimas et al. approach has been preferred since then.

Bertsimas and Sim uncertainty set

In *The Price of Robustness* [15], Bertsimas and Sim proposed a new uncertainty set that is suitable for the linear programming paradigm, even for integer variables. Given the set of uncertain data D, all defined with their central nominal values $\bar{d}$. The main concept behind this uncertainty Modeling framework is that even if lots of data coefficients are uncertain, it is a reasonable hypothesis to state that not all of them will deviate, at the same time, from their nominal value. The number of uncertain data that will deviate, and thus be set to their worst case value, is limited to Γ.

$$\mathcal{D}_\Gamma = \left\{ \Delta \in [-1,1]^{|D|} \left| \sum_{d \in D} |\Delta_d| \leq \Gamma \right. \right\}$$

As for $\mathcal{D}_\Omega$, this set is defined with only one parameter Γ. Both sets $\mathcal{D}_\Omega$ and $\mathcal{D}_\Gamma$ are suited to cases where very few information is available on the uncertainty.

Polyhedral uncertainty set

This set is quickly introduced in [7], but it is also presented as the core concept of Ben-Ameur and Kerivin's work [4] in a network design context where the traffic matrix is uncertain. It can be seen as a generalization of Bertsimas and Sim's uncertainty set $\mathcal{D}_\Gamma$ since its description is entirely linear. Therefore, that uncertainty set $\mathcal{D}_P$ can be defined as follows:

$$\mathcal{D}_P = \left\{ \Delta \in \mathbb{R}^{|D|} \left| \sum_{d \in D} a_{d,j} \Delta_d \leq b_j, \ \forall j \right. \right\}$$

This generic set is suited for problems where much more information is available on data uncertainty. As for Ben-Ameur and Kerivin, where one can estimate quite easily the future amount of traffic for given subsets of commodities, thus enabling a polyhedral description of the uncertain vector variation limitations.

1.2.4 Row-wise data uncertainty

The first case of data uncertainty that has been under study is straightforward derived from Soyster's approach. Indeed, it focuses on row-wise uncertainty. That is, given the following problem of n variables and m constraints:

$$(P_{row}) \begin{cases} \max\limits_{x} \sum\limits_{i=1}^{n} c_i x_i \\ \text{s.t.} \sum\limits_{i=1}^{n} \tilde{a}_{ij} x_i \leq b_j, \, \forall j = 1, ..., m \\ x \in X \end{cases}$$

where $\tilde{a}_{ij}$ are uncertain and X is defined as in problem (P) of section 1.2.1. Let us denote by D_j the set of uncertain data indexes in the j-th row of the linear program. To recall Soyster's approach, this would lead to solve an other deterministic problem where uncertain data is put in its worst case configuration. However, by the means of uncertainty sets presented in section 1.2.3, there is a way to reduce the conservativeness of the robust solution. This idea is common to Ben-Tal and Nemirovski [7] and Bertsimas and Sim [15]. However, since we aim to remain in the realm of linear programming, we only present Bertsimas and Sim approach in this section, but the reader should remember that a similar approach is possible for other uncertainty sets.

The main idea is, as for Soyster, to add a "protection" to the constraint, so that no matter what happens, the chosen solution remains feasible. Writing $\tilde{a}_{ij} = \bar{a}_{ij} + \hat{a}_{ij}\Delta_{ij}$ with Δ_{ij} a continuous random variable between -1 and 1, let us consider uncertainty sets $\mathcal{D}_{\Gamma}^{j}$, defined as in section 1.2.3 for all uncertain data of row j:

$$\mathcal{D}_{\Gamma_j}^{j} = \left\{ \Delta_j \in [-1, 1]^{|D_j|} \middle| \sum_{i \in D_j} |\Delta_{ij}| \leq \Gamma_j \right\}$$

Therefore, in order to have a so called robust solution, it is important that no matter what happens, the solution remains feasible. In order to do so, our uncertainty sets are only bonding uncertain data of the same row together. It is thus possible to ensure feasibility by adding a quantity in the constraint, that will act as a protection. Let us consider the robust counterpart $(P_{row}^{Bertsimas})$ of (P_{row}):

$$(P_{row}^{Bertsimas}) \begin{cases} \max\limits_{x} & \sum\limits_{i=1}^{n} c_i x_i \\ \text{s.t.} & \sum\limits_{i=1}^{n} \bar{a}_{ij} x_i + \left(\max\limits_{\Delta_j \in \mathcal{D}_{\Gamma_j}^{j}} \sum\limits_{i} \hat{a}_{ij} x_i \Delta_{ij} \right) \leq b_j, \, \forall j = 1, ..., m \quad (1.17) \\ & x \in X \end{cases}$$

In appearance, $(P_{row}^{Bertsimas})$ has an unpleasant form. Let us show how one can find his way back to the realm of linear programming.

Proposition 1.2.1. *The maximization problem* $\max_{\Delta_j \in \mathcal{D}^j_{\Gamma_j}} \sum_i \hat{a}_{ij} x_i \Delta_{ij}$ *equals the objective function of the following linear optimization problem:*

$$\beta_j(x, \Gamma_j) \begin{cases} \max\limits_{z} & \sum\limits_{i=1}^{n} \hat{a}_{ij} |x_i| z_{ij} \\ s.t. & \sum\limits_{i=1}^{n} z_{ij} \leq \Gamma_j \qquad (1.18) \\ & 0 \leq z_{ij} \leq 1, \ \forall i \in D_j \qquad (1.19) \end{cases}$$

Proof. Clearly, the optimal solution value of problem $\beta_j(x, \Gamma_j)$ consists in Γ_j variables z_{ij} at 1, and the other put to 0. This is equivalent to selecting a subset of $\Delta_{ij} \in \mathcal{D}^j_{\Gamma_j}$ variables, either put to -1 or 1. □

Which leads us to the following theorem:

Theorem 1.2.1. *Model* $(P^{Bertsimas}_{row})$ *has an equivalent linear formulation as follows:*

$$(Plinear^{Bertsimas}_{row}) \begin{cases} \max\limits_{x,y,t,p} & \sum\limits_{i=1}^{n} c_i x_i \\ s.t. & \sum\limits_{i=1}^{n} \bar{a}_{ij} x_i + t_j \Gamma_j + \sum\limits_{i \in D_j} p_{ij} \leq b_i \qquad (1.20) \\ & t_j + p_{ij} \geq \hat{a}_{ij} y_i, \ \forall j = 1, ..., m, \forall i \in D_j \qquad (1.21) \\ & -y_i \leq x_i \leq y_i, \ \forall i \qquad (1.22) \\ & x \in X \\ & p_{ij} \geq 0, \ \forall j = 1, ..., m, \forall i \in D_j \\ & y_i \geq 0, \ \forall i \\ & t_j \geq 0, \ \forall j \end{cases}$$

Proof. Let us first consider the dual of problem $\beta_j(x, \Gamma_j)$:

$$\beta^{dual}_j(x, \Gamma_j) \begin{cases} \min\limits_{t_j, p_{*j}} & \sum\limits_{i \in D_j} p_{ij} + \Gamma_j t_j \\ \text{s.t.} & t_j + p_{ij} \geq \hat{a}_{ij} |x_i|, \ \forall i \in D_j \qquad (1.23) \\ & p_{ij} \geq 0, \ \forall i \in D_j \qquad (1.24) \\ & t_j \geq 0 \qquad (1.25) \end{cases}$$

By strong duality, since problem $\beta_j(x, \Gamma_j)$ is feasible and bounded for all $\Gamma_j \in [0, |D_j|]$, then the dual problem $\beta^{dual}_j(x, \Gamma_j)$ is also feasible and bounded and their objective values coincide. Using Proposition 1.2.1, we have that problem $(P^{Bertsimas}_{row})$ is equivalent to the linear optimization problem $(Plinear^{Bertsimas}_{row})$. □

On this last proof, one should remember that the dualization of the inner maximization problem is possible and easy because in this problem $\beta_j(x, \Gamma_j)$, x variables are fixed.

Remark 1.2.1. *The robust linear optimization model* $(Plinear_{row}^{Bertsimas})$ *has* $n + k + 1$ *variables and* $m + k + n$ *constraints, where* $k = \sum_j |D_j|$ *the number of uncertain data. In most real-world applications, the constraint matrix is sparse. An attractive characteristic of the* $(Plinear_{row}^{Bertsimas})$ *formulation is that it preserves the sparsity of the constraint matrix.*

1.2.5 The specific case of right-hand-side uncertainty

In many real world applications, optimization problems with right-hand-side (RHS) uncertainty arise. As an example, for network design problems, production scheduling problems, and so on, demand uncertainty often leads to uncertain RHS formulations. Therefore, it seems incompatible with approaches presented in section 1.2.4, all based on row-wise uncertainty. This specific class of robust problem was tackled by Minoux [38, 39, 40], and in the Thesis work of Remli [46], which led to some publications with Gabrel and Murat [25, 26].

The first thing one should note is that a problem with RHS uncertainty is different from its dual counterpart. Let us show why by considering the following linear program:

$$(LP) \begin{cases} \max\limits_{x} c^T x \\ \text{s.t.} Ax \leq b \\ x \geq 0 \end{cases}$$

Now, let us define the RHS vector b as uncertain and belonging to a given uncertainty set denoted by $\mathcal{B} \subset \mathbb{R}^m$. In such case, solving the problem directly, as a special case of Soyster's model, reduces to:

$$(LP_1) \begin{cases} \max\limits_{x} c^T x \\ \text{s.t.} Ax \leq \check{b} \\ x \geq 0 \end{cases}$$

where $\check{b}_i = \min\limits_{b \in \mathcal{B}} \{b_i\}$. (LP_1) is a more restrictive deterministic problem, compared to (LP), in terms of cost.

Let us consider the dual problem of (LP_1):

$$(D_1) \begin{cases} \min\limits_{u} u^T \check{b} \\ \text{s.t.} u^T A \geq c \\ u \geq 0 \end{cases}$$

On the other hand, if we consider the dual of (LP), u being the vector of dual variables:

$$(D)\begin{cases}\min\limits_{u} & u^T b \\ \text{s.t.} & u^T A \geq c \\ & u \geq 0\end{cases}$$

Now, if we consider the robust version of (D) where b is uncertain and can take any value in the uncertainty set $\mathcal{B}$. A simple and natural objective in this context is to find u achieving a minimum value of $\max\limits_{u} u^T b$ over all possible $b \in \mathcal{B}$, thus leading to:

$$(D_2)\begin{cases}\min\limits_{u}\max\limits_{b\in\mathcal{B}} & u^T b \\ \text{s.t.} & u^T A \geq c \\ & u \geq 0\end{cases}$$

One can clearly realize that (D_1) and (D_2) are different optimization problems: they have the same solution sets, but we have $\forall u \geq 0$, $u^T \check{b} \leq \max\limits_{b\in\mathcal{B}}\{u^T b\}$, hence they have different objective optimal values.

1.2.6 Two-stage robust optimization problems

As seen in sections 1.2.4 and 1.2.5, once the problem can not be modelled with a row-wise uncertainty, one can hardly avoid the conservative aspect of robustness, inherent to Soyster's approach. Therefore, in order to tackle this issue, a new Modeling scheme was proposed by Ben-Tal and Nemirovski [6, 7], and used a lot since by many authors for many applications [1, 2, 3, 17, 40, 49]: two-stage robust optimization. The main idea behind two-stage robust optimization is similar to what is done in stochastic optimization [18]: the decision-maker will have the possibility to delay a subset of decisions in the future, after the actual scenario of uncertainty is revealed. This is also called adjustable robustness, or adaptable robustness, or even robust problem with recourse. It can be generalized to multi-stage robustness if decisions can be delayed over more than 1 time horizon.

General Modeling of two-stage robust optimization problems and theoretical results

Denoting by x and y our set of variables, let us consider a deterministic linear program:

$$(LP')\begin{cases}\min\limits_{x,y} & c^T x + d^T y \\ \text{s.t.} & Ax + By \geq b \\ & x \in X \\ & y \in Y\end{cases}$$

X and Y being sets of vectors belonging to the union of the real positive and natural set (thus making the model valid for any LP, MILP or IP problem), let us consider that some (or even all) data are uncertain in this problem, depending on a random variable ξ belonging to a bounded uncertainty set Ξ. Stating that y variables do correspond to decisions that can be delayed in the future once the realisation of the ξ vector is known for sure, let us introduce the main concepts:

Definition 1.2.1. *The two-stage linear robust optimization problem, based on (LP'), can be modelled as follows:*

$$(LP'_{robust}) = \begin{cases} \min\limits_{x\in X} \max\limits_{\xi\in\Xi} \; c^T x + \min\limits_{y\in Y} \; d^T y \\ \qquad\qquad\qquad\qquad s.t. \;\; By \geq b - Ax \end{cases}$$

Remark 1.2.2. *For the Modeling of (LP'_{robust}), the worst case criterion has been chosen (see section 1.2.2), as it is mostly done in the literature.*

Remark 1.2.3. *This model is valid for any set of uncertain data. That includes the previously examined row-wise uncertainty case, and the RHS uncertainty one.*

Definition 1.2.2. *If, for any valid solution vector x, and any $\xi \in \Xi$, there always exists at least one feasible solution vector $y(x, \xi)$, then the **full recourse property** is verified.*

Definition 1.2.3. *The maximization problem that is contained in the objective function of problem (LP') is often referred as **the recourse problem** and is denoted by*

$$\mathcal{Q}(x) = \max_{\xi\in\Xi} \; c^T x + \min_{\left(\begin{array}{c} y \in Y \\ By \geq b - Ax \end{array} \right)} d^T y$$

Of course, as such, the problem (LP'_{robust}) is not linear. However, there exists a general linearisation scheme for this problem:

Proposition 1.2.2. *The problem (LP'_{robust}) can be modelled as follows:*

$$(LPlinear'_{robust}) \begin{cases} \min \; \eta & \\ \quad s.t. \;\; \eta \geq c^T(\xi)x + d^T(\xi)y(\xi), & \forall \xi \in \Xi \\ \qquad A(\xi)x + B(\xi)y(\xi) \geq b(\xi), & \forall \xi \in \Xi \\ \qquad x \in X & \\ \qquad y(\xi) \in Y, & \forall \xi \in \Xi \end{cases}$$

Of course, $(LPlinear'_{robust})$ is almost infinite in size, depending on the uncertainty set Ξ. Note that the notation $y(\xi)$ means that in $(LPlinear'_{robust})$, there is a different set of recourse variables y for each possible scenario $\xi \in \Xi$.

Theorem 1.2.2. *The problem* $(LPlinear'_{robust})$ *with RHS uncertainty (that is, only the vector* b *is uncertain) and* Ξ *being a finite bounded polyhedron (like Bertsimas and Sim's uncertainty set) is strongly NP-hard, even for continuous* x *and* y *variables.*

Proof. See the work of Minoux [40]. □

A few works have focused on 2 stage robust problems to compare the single stage approach to the 2 stage one. The reader may refer to [11] for more information. There is also a well-known application of 2 stage robust optimization to network design problems done by Atamtürk et al. [1].

An exact solving approach for the 2 stage robust problem with continuous recourse and RHS uncertainty

Since 2 stage robust problems are often NP-hard, there is a few works available on how to solve specific versions of that problem. Thiele et al. [49], Zeng et al. [54] and Billionnet et al. [17] proposed a generic method for solving 2 stage robust problems with continuous recourse variables in the case of RHS uncertainty. Let us write this problem:

$$(LP^{RHS}_{robust}) \begin{cases} \min\limits_{x\in X} \max\limits_{\xi\in\Xi} \min\limits_{y\geq 0} \; c^T x + d^T y \\ \qquad\qquad\qquad \text{s.t. } \; By \geq b(\xi) - Ax \end{cases}$$

Which can be linearized as follows:

$$(LPlinear^{RHS}_{robust}) \begin{cases} \min \; \eta & \\ \text{s.t. } \; \eta \geq c^T x + d^T y(\xi), & \forall \xi \in \Xi \\ \qquad Ax + By(\xi) \geq b(\xi), & \forall \xi \in \Xi \\ \qquad x \in X & \\ \qquad y(\xi) \geq 0, & \forall \xi \in \Xi \end{cases}$$

Note that, as mentioned before, $(LPlinear^{RHS}_{robust})$ may be infinite in size. Here, authors studied the problem with the Bertsimas and Sim's uncertainty set. However, as showed by Minoux in [39, 40], this problem is strongly NP-hard and writing the whole problem directly may prove intractable in practice. Therefore, authors proposed a column-and-constraint (because in addition to new constraints, a set of recourse variables y must be generated for each possible ξ) generation algorithm which is presented in a short version in this bibliography. In all the following, the full recourse property is assumed to be verified. However, note that for cases that do not include this property, Billionnet et al. [17] proposed a generalization of the described method.

Proposition 1.2.3. *For any S_Ξ being a subset of the set of extremal points of the polyhedron Ξ, the following problem gives a lower bound for the problem $(LPlinear^{RHS}_{robust})$:*

$$(LPlinear^{RHS}_{S_\Xi}) \begin{cases} \min \ \eta & \\ s.t. \ \eta \geq c^T x + d^T y(\xi), & \forall \xi \in S_\Xi \\ \quad Ax + By(\xi) \geq b(\xi), & \forall \xi \in S_\Xi \\ \quad x \in X & \\ \quad y(\xi) \geq 0, & \forall \xi \in S_\Xi \end{cases}$$

Note that this is true for any subset of Ξ as well, S_Ξ being a specific case.

Proof. $(LPlinear^{RHS}_{S_\Xi})$ is actually a relaxed version of $(LPlinear^{RHS}_{robust})$ where column and constraints for $\xi \in \Xi \backslash S_\Xi$ are not taken into account. Since the cost is only supported by the first set of constraints, adding new columns will either change nothing to the value of η, or it will increases this value. Therefore it gives a lower bound of the global problem. □

Proposition 1.2.4. *For any feasible solution vector x', solving the corresponding recourse problem $\mathcal{Q}(x')$ gives an upper bound for the problem.*

Proof. Once the full recourse property established, this result is immediate since every solution of a minimizing problem gives an upper bound for this problem. □

From here, it is theoretically possible to find both lower and upper bounds for the problem. However, $\mathcal{Q}(x)$ remains to be solved.

Theorem 1.2.3. *The exact value of the recourse problem $\mathcal{Q}(x)$ can be computed by solving the following quadratic problem:*

$$\mathcal{Q}(x) \begin{cases} \max\limits_{\lambda,\xi} c^T x + \lambda^T \left(b(\xi) - Ax\right) & \\ \quad s.t. \lambda^T B \leq d & \\ \quad\quad \xi \in \Xi & \\ \quad\quad \lambda \geq 0 & \end{cases}$$

Proof. The result is obtained by dualizing the inner minimization problem of the recourse problem. Since y variables are continuous, strong duality holds. □

Remark 1.2.4. *It is important to note what the **full recourse property** says and what it does not. Indeed, having the full recourse property verified guarantees that solving $(LPlinear^{RHS}_{S_\Xi})$ and $\mathcal{Q}(x)$ will always be feasible. However, if the property is not verified, it does not mean that these problems will become infeasible: it just removes the guarantee. Various issues can raise when the property is not verified. The main one is that for a*

given solution x, no solution y can be found for any $\xi \in \Xi$, that is: $\mathcal{Q}(x)$ is infeasible. But that does not imply that the global problem ($LPlinear^{RHS}_{robust}$) is infeasible as well since other values of x would have been feasible.

Of course, since b is actually depending on ξ, there will be some quadratic terms in the objective function. However, authors showed that while working with Bertsimas and Sim's uncertainty set, ξ variables have a boolean behaviour, thus making the linearization possible. Hence, by alternatively solving ($LPlinear^{RHS}_{S_\Xi}$) for a reduced set of scenarios S_Ξ, it is possible to solve the recourse problem for the solution vector x' found. Solving $\mathcal{Q}(x')$ will give an upper bound and, more importantly, a new scenario vector ξ' (which is the worst one for x') to add to the set S_Ξ. Authors shows that by repeating this process, convergence is assured and in practice, only a small set (in comparison to the whole Ξ) is necessary to solve ($LPlinear^{RHS}_{robust}$) to optimality. Algorithm 1 gives a sketch of that procedure.

Algorithm 1 ($LPlinear^{RHS}_{robust}$) solving procedure

Require: S_Ξ, UB, LB

1: **while** $UB - LB > 0$ **do**
2: solve ($LPlinear^{RHS}_{S_\Xi}$) and update LB
3: solve $\mathcal{Q}(x')$
4: $UB \leftarrow \min\{UB, \mathcal{Q}(x')\}$
5: $S_\Xi \leftarrow S_\Xi \cup \{\xi'\}$
6: **end while**

Affine decision rules on recourse variables for solving 2 stage robust problems

Since 2 stage robust problems can be very large in size, it may be intractable in practice to generate only "good" scenarios by the means of a procedure similar to the one presented in the last section 1.2.6. In order to tackle this issue of a potentially infinite number of recourse variables, Ben-Tal and Nemirovski first proposed an alternative in [6, 7] named the Affinely Adjustable Robust Counterpart. This work was further described by Babonneau, Vial and Apparigliato [2]. Applications of this approach have been done, particularly in network design problems by Ouorou [43], Babonneau, Klopfenstein, Ouorou and Vial [3]. An in-depth study of the existing relations between the initial 2 stage robust problem and its affinely adjustable robust counterpart has been done by Bertsimas and Goyal [11], Bertsimas, Iancu and Parrilo [13], Iancu, Sharma and Sviridenko [32]. In these works, they identify a few cases where the affine decision rule is actually optimal.

Let us look back to our generic model for 2 stage robust optimization problems, given by definition 1.2.1:

$$(LP'_{robust}) = \begin{cases} \min\limits_{x\in X} \max\limits_{\xi\in\Xi} \; c^T x + \min\limits_{y\in Y} \; d^T y \\ \qquad\qquad\qquad\qquad \text{s.t. } By \geq b - Ax \end{cases}$$

Which can be rewritten in a linear way as:

$$(LPlinear'_{robust}) \begin{cases} \min \; \eta & \\ \text{s.t. } \eta \geq c^T(\xi)x + d^T(\xi)y(\xi), & \forall \xi \in \Xi \\ \quad A(\xi)x + B(\xi)y(\xi) \geq b(\xi), & \forall \xi \in \Xi \\ \quad x \in X & \\ \quad y(\xi) \in Y, & \forall \xi \in \Xi \end{cases}$$

The trick here is to consider optimal recourse decisions y as functions of the uncertain vector ξ: $y(\xi)$. Of course, an analytical expression of that function is not available. Even thinking about the shape, continuity, of this function can prove irrelevant, especially for discrete y. The main idea of this approach is to approximate this unknown function by an affine function. This can also be seen as a Taylor development of the first order-like of this function.

Hypothesis 1. *We assume that any recourse variable y can be written as an affine function of the uncertain vector ξ. Thus we shall write, for a given set of uncertain data D:*

$$y = y_0 + \sum_{h\in D} y_h \xi_h$$

This approximation, as mentioned before, leads to a problem that is different from the original one, called the Affinely Adjustable Robust Counterpart (or AARC). We use index h for the sake of clarity because the uncertainty set includes all uncertain data of the problem, which are not necessarily in a row-wise or column-wise repartition. The number of first stage variables x is denoted by n and the number of recourse variables y is denoted by n'. Let us rewrite $(LPlinear'_{robust})$ under hypothesis 1, denoting this new problem $(LPlinear'_{AARC})$:

$$(LPlinear'_{AARC}) \begin{cases} \min\ \eta & \\ \text{s.t. } \eta \geq \sum_{i=1}^{n} c_i(\xi)x_i + \sum_{i'=1}^{n'} d_{i'}(\xi)\left(y_{i',0} + \sum_{h\in D} y_{i',h}\xi_h\right), & \forall \xi \in \Xi \\ \sum_{i=1}^{n} a_{ij}(\xi)x_i + \sum_{i'=1}^{n'} b_{i'j}(\xi)\left(y_{i',0} + \sum_{h\in D} y_{i',h}\xi_h\right) \geq b_j(\xi), & \forall \xi \in \Xi, \forall j \\ x \in X & \\ y_{i',0} + \sum_{h\in D} y_{i',h}\xi_h \geq 0, & \forall i', \forall \xi \in \Xi \\ y_{i',0} \geq 0, & \forall i' \\ y_{i',h} \in \mathbb{R}, & \forall i', \forall h \end{cases}$$

Without loss of generality, we can replace the generic $y(\xi) \in Y$ by $y_{i',0} + \sum_{h\in D} y_{i',h}\xi_h \geq 0$ because, once put in the affine form, it does not matter whether $y(\xi)$ was actually integer or not. Now, depending on what data are actually uncertain, solving this problem will prove more or less easy. The literature mainly focuses on the so called **static recourse problem** case.

Definition 1.2.4. *The **static recourse property** is verified when the data related to the recourse matrix and the recourse cost is not uncertain.*

It can be put in another way by stating that all data corresponding to recourse variables y are not uncertain. In $(LPlinear'_{AARC})$, it would mean, for the static recourse property to be verified, that $d_{i'}$ (respectively $b_{i'j}$) must not be uncertain for any i' (respectively for any $i', \forall j$).

When the static recourse property holds, the problem can be rewritten as follows:

$$(LPstatic'_{AARC}) \begin{cases} \min\ \eta & \\ \text{s.t. } \eta \geq \sum_{i=1}^{n} c_i(\xi)x_i + \sum_{i'=1}^{n'} d_{i'}\left(y_{i',0} + \sum_{h\in D} y_{i',h}\xi_h\right), & \forall \xi \in \Xi \\ \sum_{i=1}^{n} a_{ij}(\xi)x_i + \sum_{i'=1}^{n'} b_{i'j}\left(y_{i',0} + \sum_{h\in D} y_{i',h}\xi_h\right) \geq b_j(\xi), & \forall \xi \in \Xi, \forall j \\ x \in X & \\ y_{i',0} + \sum_{h\in D} y_{i',h}\xi_h \geq 0, & \forall i', \forall \xi \in \Xi \\ y_{i',h} \in \mathbb{R}, & \forall i', \forall h = 0, ..., m \end{cases}$$

In order to remain concise yet clear enough on the method, let us restrain ourselves in this bibliography to a subset of 2 stage robust problems with static recourse. The point

being to give the reader a general idea of the method so he can extend it to any static recourse case. Thus, let us assume that we are dealing with a case of RHS uncertainty (as in Ouorou's work [43] for example, where a network design problem under demand uncertainty is tackled). The uncertainty set is the Bertsimas and Sim's one. Therefore, only RHS b_j values are uncertain, and written as follows:

$$b_j(\xi) = \bar{b}_j + \hat{b}_j \xi_j$$

and

$$\Xi_\Gamma = \left\{ \xi \in [-1,1]^{|D|} \left| \sum_h \xi_h \leq \Gamma \right. \right\}$$

with $D = 1, ..., m$. Let us write the problem:

$$(LPrhs'_{AARC}) \begin{cases} \min \ \eta & \\ \text{s.t. } \eta \geq \displaystyle\sum_{i=1}^{n} c_i x_i + \sum_{i'=1}^{n'} d_{i'} y_{i',0} + \sum_{i'=1}^{n'} d_{i'} \sum_{h=1}^{m} y_{i',h} \xi_h, & \forall \xi \in \Xi_\Gamma \\ \displaystyle\sum_{i=1}^{n} a_{ij} x_i + \sum_{i'=1}^{n'} b_{i'j} y_{i',0} + \sum_{i'=1}^{n'} b_{i'j} \left(\sum_{h=1}^{m} y_{i',h} \xi_h \right) - \hat{b}_j \xi_j \geq \bar{b}_j, & \forall \xi \in \Xi_\Gamma, \forall j \\ x \in X & \\ y_{i',0} + \displaystyle\sum_{h=1}^{m} y_{i',h} \xi_h \geq 0, & \forall i', \forall \xi \in \Xi_\Gamma \\ y_{i',h} \in \mathbb{R}, & \forall i', \forall h = 0, ..., m \end{cases}$$

Proposition 1.2.5. *The problem $(LPrhs'_{AARC})$ can be equivalently modelled as follows:*

$$(LPrhs''_{AARC}) \begin{cases} \min \ \eta & \\ s.t. \ \eta \geq \displaystyle\sum_{i=1}^{n} c_i x_i + \sum_{i'=1}^{n'} d_{i'} y_{i',0} + \max_{\xi \in \Xi_\Gamma} \left(\sum_{i'=1}^{n'} d_{i'} \sum_{h=1}^{m} y_{i',h} \xi_h \right) & \\ \displaystyle\sum_{i=1}^{n} a_{ij} x_i + \sum_{i'=1}^{n'} b_{i'j} y_{i',0} + \min_{\xi \in \Xi_\Gamma} \left(\sum_{i'=1}^{n'} b_{i'j} \left(\sum_{h=1}^{m} y_{i',h} \xi_h \right) - \hat{b}_j \xi_j \right) \geq \bar{b}_j, & \forall j \\ x \in X & \\ y_{i',0} + \min_{\xi \in \Xi_\Gamma} \left(\displaystyle\sum_{h=1}^{m} y_{i',h} \xi_h \right) \geq 0, & \forall i' \\ y_{i',h} \in \mathbb{R}, & \forall i', \forall h = 0, ..., m \end{cases}$$

Proof. Constraints of problem $(LPrhs'_{AARC})$ must be verified for all possible ξ in Ξ_Γ. Therefore, depending on the side of the constraint that random variables are in, each constraint thus must be verified for the *worst* value the uncertain quantity can take (either maximal or minimal). □

Proposition 1.2.5 enables to find a very similar problem to the one studied by Bertsimas and Sim presented in section 1.2.4.

Theorem 1.2.4. *($LPrhs''_{AARC}$) can be solved by solving the following linear program:*

$$
(LPrhs'''_{AARC}) \begin{cases}
\min\ \eta & & \\
s.t.\ \eta \geq \sum_{i=1}^{n} c_i x_i + \sum_{i'=1}^{n'} d_{i'} y_{i',0} + \Gamma t^0 + \sum_{h=1}^{m} p_h^0 & & \\
\quad -(t^0 + p_h^0) \leq \sum_{i'=1}^{n'} d_{i'} y_{i',h} \leq t^0 + p_h^0 & & \forall h \\
\quad \sum_{i=1}^{n} a_{ij} x_i + \sum_{i'=1}^{n'} b_{i'j} y_{i',0} + \Gamma t^j + \sum_{h=1}^{m} p_h^j \geq \bar{b}_j & & \forall j \\
\quad -(t^j + p_j^j) \leq \sum_{i'=1}^{n'} b_{i'j} y_{i',j} - \hat{b}_j \leq t^j + p_j^j & & \forall j \\
\quad -(t^j + p_h^j) \leq \sum_{i'=1}^{n'} b_{i'h} y_{i',h} \leq t^j + p_h^j & & \forall j, \forall h \neq j \\
\quad x \in X & & \\
\quad y_{i',0} + \Gamma t^{i'} + \sum_{h=1}^{m} p_h^{i'} \geq 0 & & \forall i' \\
\quad -(t^{i'} + p_h^{i'}) \leq y_{i',h} \leq t^{i'} + p_h^{i'}, & & \forall i', \forall j \\
\quad y_{i',h} \in \mathbb{R} & & \forall i', \forall h = 0, ..., m \\
\quad t^h \geq 0 & & \forall h = 0, ..., m \\
\quad t^{i'} \geq 0 & & \forall i' \\
\quad p_h^j \geq 0 & & \forall j = 0, ..., m, \forall h \\
\quad p_h^{i'} \geq 0 & & \forall h, \forall i'
\end{cases}
$$

Proof. This equivalent, finite, linear Modeling is found by dualizing every inner optimization problem in ξ. As ξ is continuous, strong duality holds as the Bertsimas and Sim uncertainty set Ξ_Γ has a polyhedral description. The reader may refer to section 1.2.4 for more details. □

Therefore, by making the affine approximation on recourse variables, it is thus possible to transform the model into a row-wise uncertainty case. Then, all that remains to be done is applying the usual dualization technique to have a good looking linear program. However, one should remember that this transformation remains greedy in terms of additional columns and constraints. Of course, this is all an approximation, and the reader should keep in mind that we do not solve the same problem at all. But the most

important part of the decisions is the first stage ones (the x variables in our model). The Affinely Adjustable Robust Counterpart mainly aims to find "good" first stage decisions.

The K-adaptable 2 stage robust problem

This section focuses on an alternative way of Modeling robust problems. It has been developed by Bertsimas and Caramanis in a general context [10] (2010) and in parallel by Ben-Ameur and Zotkiewicz for the robust routing problem in particular [5] (2011). This approach can be seen as a generalization of the usual robust approach. Indeed, it aims at bridging the gap between a single stage robust problem and a 2 stage robust problem (it can be extended to multi stage robust problems). They pointed out the fact that only 2 ways of acting against uncertainty are available in the robustness literature, that is:

- Deciding *before* the actual scenario is revealed, thus *never* adapting the solution to this scenario.
- Having a subset of decisions that can be taken *after* the actual scenario is revealed, thus *always* adapting the solution to that scenario.

Authors claimed that both may prove irrelevant in some cases. As the single stage approach is often viewed as too pessimistic and too conservative, the 2 stage approach which states that the decision-maker may be able to adapt himself to every possible scenario could be too optimistic too (and hardly tractable in many cases, especially when the recourse problem wields discrete variables). Indeed, one can easily conceive that for some heavy organization, a quick reactivity to uncertainty may be impossible to put in practice. However, having a reduced set of K prepared responses to the uncertainty, thus limiting the decision to "which one should we pick?" can make sense in such context. Therefore, they both proposed a new kind of robust models where the decision-maker will not be able to adapt himself to all scenarios, while ensuring robust feasibility, thus proposing an alternative solution that is really *between* single stage and 2 stage robust Modeling.

Let us recall the general model for 2 stage robust optimization problems:

$$(LP'_{robust}) = \begin{cases} \min\limits_{x\in X} \max\limits_{\xi\in\Xi} \min\limits_{y\in Y} \; c^T x + d^T y(\xi) \\ \qquad\qquad\qquad \text{s.t. } B(\xi)y \geq b - A(\xi)x \end{cases}$$

Note that without loss of generality, c and d vectors can be considered not uncertain, as an uncertain cost function can always be put in the constraint matrix. In this problem, adjustable decision variables y can be of different values for any possible $\xi \in \Xi$. On the

opposite, the single stage version of that problem which forbids any possible adaptation to the outcome is:

$$(LP'_{static}) = \begin{cases} \min\limits_{x\in X, y\in Y} c^T x + d^T y \\ \quad \text{s.t. } A(\xi)x + B(\xi)y \geq b, \ \forall \xi \in \Xi \end{cases}$$

In the proposed K-adaptability problem, the decision-maker aims to define K adjustable solutions $\{y_1, y_2, ..., y_K\}$ (that will be found by the optimization problem), and then commits to one of them only after seeing the realization of the uncertainty. Of course, at least one of the K solutions must be feasible regardless of the realization of the uncertainty. Authors thus defined the problem (LP'_{adapt}) as follows:

$$(LP'_{adapt}) = \begin{cases} \min\limits_{x, y_1, ..., y_K} & c^T x + \max\left\{d^T y_1, d^T y_2, ..., d^T y_k\right\} \\ \text{s.t.} & \begin{bmatrix} A(\xi)x + B(\xi)y_1 \geq b \\ or \\ A(\xi)x + B(\xi)y_2 \geq b \\ or \\ ... \\ or \\ A(\xi)x + B(\xi)y_k \geq b \end{bmatrix} \quad \forall \xi \in \Xi \end{cases}$$

Authors then show that this problem can be reduced to a K-partition problem where one has to find the best way of partitioning the uncertainty set Ξ into K uncertainty subsets, so that $\Xi = \Xi_1 \cup \Xi_2 \cup ... \cup \Xi_K$. Then it is possible to rewrite (LP'_{adapt}) as such:

$$\min_{\Xi = \Xi_1 \cup \Xi_2 \cup ... \cup \Xi_K} \begin{cases} \min\limits_{x, y_1, ..., y_K} & c^T x + \max\left\{d^T y_1, d^T y_2, ..., d^T y_k\right\} & \\ \text{s.t.} & A(\xi)x + B(\xi)y_1 \geq b & \forall \xi \in \Xi_1 \\ & A(\xi)x + B(\xi)y_2 \geq b & \forall \xi \in \Xi_2 \\ & ... & \\ & A(\xi)x + B(\xi)y_k \geq b & \forall \xi \in \Xi_k \end{cases}$$

Once the optimal K-partition is found, solving (LP'_{adapt}) can be done in a tractable way if (LP'_{static}) was tractable as well. Note that this model works for discrete recourse variables y. However, by doing so, the difficulty of the problem has been shifted to the finding of the optimal K partition for the problem. In terms of complexity, Bertsimas and Caramanis showed that even for a 2-partition of the uncertainty set for RHS uncertainty where Ξ is a simple polyhedron the one defined by Bertsimas and Sim is NP-hard.

1.3 Other interesting theoretical tools or methods used in this study

This section contains short summaries of widely used methods, amongst other well known definitions, that any reader may need while reading the present work. Topics presented

here were considered to be too generic, or too simple to have their own dedicated section.

1.3.1 The Newton-Raphson method

Let us just recall the basic principle of the Newton-Raphson method. It is an iterative process that can be used to find roots of a real-valued function. Given a function f that can be derived over $I \subseteq \mathbb{R}$, the method provides a series of points x_i that tends to approximate a root of the function, denoted by x_r.

For any $x_0 \in I$ such that $f'(x_0) \neq 0$, the series is defined as follows:

$$\begin{aligned} x_0 &\in I \\ x_{i+1} &= x_i - \frac{f(x_i)}{f'(x_i)}, \ \forall i \geq 1 \end{aligned}$$

Many things could be said on how the method converges and how the issue of having multiple roots could be handled, but as it will not be needed in this study, me leave that aside. That short section is just a quick reminded for the reader.

1.3.2 The Ordered Weighted Average in multi-criteria optimization

A multi-criteria optimization problem (PMC) is an optimization problem in which several criteria must be optimized simultaneously. Given x the set of variables, it can be formulated as follows:

$$(PMC) \begin{cases} \max\limits_{x} \ (c_1(x), ..., c_m(x)) \\ \text{s.t. } x \in X \end{cases}$$

with $X \subseteq \mathbb{R}^n$.

To tackle such problems, many options are available, such as finding all Pareto-optimal solutions and let the decider pick amongst them those he prefer. However, the set of Pareto-optimal solutions may be huge and the decider's task may be difficult to complete.

Thus, an other way for tackling multi-criteria optimization is to use an aggregation operator that will combine all criteria into a well chosen aggregate in order to optimize it as a single criterion. The first that comes to mind usually is the arithmetic mean: to each criterion c_i, a weight $\mu_i \geq 0$ is associated with $\sum_{i=1}^{m} \mu_i = 1$ (without loss of generality). Thus we can define a new problem (PMC_M) by using the arithmetic mean as an aggregation operator:

$$(PMC_M) \begin{cases} \max\limits_{x} M(x) \\ \text{s.t. } x \in X \end{cases}$$

with $M(x) = \sum_{i=1}^{m} \mu_i c_i(x)$.

Other operators can be considered, such as the Ordered Weighted Average operator (OWA). Here, weights are also associated to criteria, but in an increasing order. Given a decreasing sequence of weights $\lambda_i \geq 0$ such that $\sum_{i=1}^{m} \lambda_i = 1$. Criteria are sorted in an increasing order:

$$c_{(1)}(x) \leq ... \leq c_{(m)}(x)$$

where (.) is the criteria permutation that place the smallest one in first, the second smallest one after, and so on until the biggest one is set at the end of the sequence. Thus we can define a new problem (PMC_{OWA}) by using the ordered weighted average as an aggregation operator:

$$(PMC_{OWA}) \begin{cases} \max\limits_{x} OWA(x) \\ \text{s.t. } x \in X \end{cases}$$

with $OWA(x) = \sum_{i=1}^{m} \lambda_i c_{(i)}(x)$.

Note that if $\lambda_1 = \lambda_2 = ... = \lambda_m$, then $OWA(x) = M(x)$ where all μ_i coefficients are also equal. Moreover, if $\lambda_1 = 1$ and $\lambda_i = 0$ for all $i \geq 2$, then $OWA(x) = \min\limits_{i=1,...,m} c_i x$.

In order to model the OWA criterion in a linear program, one can easily check that the permutation that sort criteria in an increasing order is the one that minimizes $\sum_{i=1}^{m} \lambda_i c_{(i)}(x)$. Thus, in order to find the proper criteria permutation, one needs to solve the following affectation problem for a given x:

$$(AFF) \begin{cases} \min\limits_{y} \sum\limits_{j=1}^{m} \lambda_j \left(\sum\limits_{i=1}^{m} c_i(x) y_{ij} \right) \\ \text{s.t. } \sum\limits_{i=1}^{m} y_{ij} = 1, \forall j = 1, ..., m \\ \quad \sum\limits_{j=1}^{m} y_{ij} = 1, \forall i = 1, ..., m \\ \quad y \in \mathbb{R}_{+}^{m\times} \end{cases}$$

Note that y variables are considered continuous since they automatically take integer values in the linear program (AFF). Thus, $y_{ij} = 1$ if, and only if the criterion c_i is affected to the weight λ_j. However, in practice, criteria c_i depends on x and, as such, (AFF) could lead to quadratic formulations. In order to have a linear program, it is

necessary to consider the dual of the affectation problem:

$$(DAFF)\begin{cases} \max\limits_{\alpha,\beta} \sum\limits_{j=1}^{m} \alpha_i + \beta_i \\ \text{s.t. } \alpha_i + \beta_j \leq \lambda_j c_i(x),\ \forall i = 1, ..., m, \forall j = 1, ..., m \\ \quad \alpha \in \mathbb{R}^m, \beta \in \mathbb{R}^m \end{cases}$$

Thus, each criterion is multiplied by a constant instead of a variable. Note that variables α and β have no sign here. For example, if one needs to maximize $OWA(x)$, it can be done by solving:

$$(PMC_{OWA})\begin{cases} \max\limits_{x,\alpha,\beta} \sum\limits_{j=1}^{m} \alpha_i + \beta_i \\ \text{s.t. } \alpha_i + \beta_j \leq \lambda_j c_i(x),\ \forall i = 1, ..., m, \forall j = 1, ..., m \\ \quad x \in X \\ \quad \alpha \in \mathbb{R}^m, \beta \in \mathbb{R}^m \end{cases}$$

The OWA aggregation operator has a few useful properties that are listed below.

Property 1.3.1. *Given m criteria $c_1, ..., c_m$ such that $0 \leq c_{(1)} \leq ... \leq c_{(m)}$ where $(.)$ is the permutation that sort them in an increasing order. Let us have two ordered weighed averages OWA_μ and OWA_λ of these m criteria with $\mu_1 \geq \lambda_1$ and $\mu_i \leq \lambda_i$ for $2 \leq i \leq m$. Therefore, $OWA_\mu \leq OWA_\lambda$.*

Property 1.3.2. *Given m criteria $c_1, ..., c_m$ such that $0 = c_{(1)} = ... = c_{(k)} < c_{(k+1)} \leq ... \leq c_{(m)}$ with $1 \leq k \leq m-1$. Let us have two ordered weighed averages OWA_μ and OWA_λ of these m criteria with $\mu_1 \geq \lambda_1$ and $\mu_i \leq \lambda_i$ for $2 \leq i \leq m$ with at least an index $j \geq k+1$ for which $\mu_j < \lambda_j$. Therefore, $OWA_\mu < OWA_\lambda$.*

Proof. $OWA_\mu - OWA_\lambda = \sum\limits_{i=k+1}^{m} (\mu_i - \lambda_i) c_{(i)} < 0$ □

Chapter 2

Integration of Organisation, Administration & Management considerations in the passive optical network deployment optimization problem

Introduction and objectives

For long, network design strategies have been driven by mere deployment capital expenditures (CAPEX, which can generally be seen as the "here and now" cost of the network, in terms of installed equipments and required workforce). However, major sources of costs on the long term are not the direct CAPEX costs but indirect and recurrent costs linked to Operations, Administration and Maintenance (OA&M) which are part of what is usually called Operational Expenditures (OPEX, in opposition to CAPEX). For a comprehensive survey on network administration and maintenance concepts, the reader is referred to [19]. This includes, but is not limited to, Information System costs, monitoring and supervision costs, preventive and curative maintenance costs. Thus, important research efforts are being spent on the field of Autonomics, i.e. roughly speaking the design of networks and protocols able to perform such functionalities by themselves, with the aim of getting rid of these costs. These are obviously long-term objectives. Nevertheless, it is not going too far to acknowledge that telecommunication operators are highly concerned about facilitating future OA&M costs of their network when discussing network design strategies and operational deployment schemes.

As shown in the bibliographical study of section 1.1, there are few works associating

both CAPEX and OPEX costs in a single optimization problem, whatever the problem is, and so there is nothing that has been done on the specific FTTH PON deployment optimization problem about this issue. This is mainly due to the fact that CAPEX and OPEX optimization problems are often difficult both in terms of computational complexity and computational tractability, even when we tackle them separately, so treating them both at the same time may lead to an even greater difficulty on those aspects. But the most blocking issue is that those costs are not of the same dimension. Indeed, CAPEX are here-and-now costs while OPEX are overtime costs. Consequently, we would be dealing with multi criteria optimization over very complex problems.

In this chapter, we aim to take these OA&M considerations into account for optimizing the deployment of a Passive Optical Network. However, since these future OA&M costs are very diverse, and for all the reasons mentioned above, we decided to tackle this issue by adding them to our models as engineering rules. That is, adding them as constraints, because we aim at having a beneficial effect on future OA&M costs, even if they increase the CAPEX deployment costs. More than the inherent great difficulty of the problem that we aim to avoid, there is also a very practical and telecommunication company related reason to do so. In fact, in most telecommunications operators companies that are in charge of deploying a network and, later, exploiting it, entities in charge of each tasks are often not the same, which means they do not run their activities on the same budget and they do not have the same objectives. One can easily conceive that deployment teams will consider their job done if the network deployment was made the cheapest way possible, while network monitoring teams will have to do their best with the network they will be given. Moreover, primary clients for such optimization models are deployment teams, and the reality of a nowadays company being as it is, each part of it trying to get as much budget as possible, it could be really hard to find a harmony between those two very different costs if they were to be optimized at the same time over their Pareto fronts.

The objective of this chapter is to identify engineering rules that may be easily (which must be understood as easily pluggable into our model) be added to the FTTH PON deployment problem, without changing its cost objective (but not its value), to analyse their impact in terms of cost and tractability of the problem, and finally, to propose a way of assessing quantitatively and qualitatively the future gain one can expect after adding these engineering rules to a FTTH PON model. Note that engineering rules will not be exhaustive for this problem: two types of rules will be considered, drawn from field considerations. Our second objective is to propose a preliminary work on this issue, while trying to initiate what we think to be a "good" optimization practice which may lead, for some problem, to interesting research. The work exposed in this chapter has

been published in 2012 [1].

2.1 The choice of engineering rules in an OA&M perspective

In this section, we aim to propose various engineering rules. For that purpose, we will focus on the general FTTH PON deployment optimization problem described in section 1.1.1. We first generalize the (PON^1) and (PON^2) models presented in section 1.1.2 to a K level of splitters PON architecture. We use the same notations as in the previous section.

$$
(PON^K)\begin{cases}
\min\limits_{\mathbf{f},\mathbf{z},\mathbf{u}} & \sum\limits_{i=1}^{n}\sum\limits_{k=1}^{K} C^k \mathbf{z}_i^k + \sum\limits_{(i,j)\in E}\sum\limits_{k=1}^{K+1} c_{ij}^k \left(\mathbf{f}_{ij}^k + \mathbf{f}_{ji}^k\right) & \\
\text{s.t.} & \sum\limits_{j\neq i} \mathbf{f}_{ji}^1 = \mathbf{z}_i^1 + \sum\limits_{j\neq i} \mathbf{f}_{ij}^1,\ \forall i \in V^* & (2.1)\\
& \sum\limits_{j\neq i} \mathbf{f}_{ji}^k + m^{k-1}\mathbf{z}_i^{k-1} = \mathbf{z}_i^k + \sum\limits_{j\neq i} \mathbf{f}_{ij}^k + \mathbf{u}_i^k,\ \forall k = 2,...,K, \forall i \in V & (2.2)\\
& \sum\limits_{j\neq i} \mathbf{f}_{ji}^{K+1} + m^K \mathbf{z}_i^K = d_i + \sum\limits_{j\neq i} \mathbf{f}_{ij}^{K+1} + \mathbf{u}_i^{K+1},\ \forall i \in V & (2.3)\\
& \sum\limits_{k=1}^{K+1} \left(\mathbf{f}_{ij}^k + \mathbf{f}_{ji}^k\right) \leq b_{ij},\ \forall (i,j) \in E & (2.4)\\
& \mathbf{z}_i^k \in \mathbb{N},\ \ \forall i \in V^*; k = 1,...,K; \mathbf{z}_0^k = 0, \forall k = 1,...,K & \\
& \mathbf{u}_i^k \in \mathbb{N},\ \ \forall i \in V^*; k = 2,...,K+1; \mathbf{u}_0^k = 0, \forall k = 2,...,K & \\
& \mathbf{f}_{ij}^k \in \mathbb{N},\ \ \forall (i,j) \in E, k = 1,...,K+1 &
\end{cases}
$$

Note that in practice, in recorded PON architecture deployed so far, the number of optical splitters does not exceed 3, as optical splitters will consume a high amount of optical budget, as explained in section 1.1.2.

2.1.1 Modeling of the optical splitter delocation rule

For the sake of simplicity in future network administration (and notably when thinking of fault monitoring and troubleshooting activities), network managers do appreciate that subscribers of the same building (or neighbourhood) have the same "connection point" to the network. Likewise, from a customer relationship point of view, managers

[1] C. Hervet, M. Chardy, *Passive optical network design under operations administration and maintenance considerations*, Journal of Applied Operational Research, Vol. 4(3), pp. 152–172, 2012

do appreciate that subscribers experience the same Quality of Service, as it is always complicated to explain to two neighbours affording the "same" subscription that they are not delivered the "same" service. This is classically achieved by mono-routing strategies in multi flow-based network design problems. Such reasoning can be extended to the PON context, meaning that both optical routes and location of splitters (of any level) are similar for the subscribers of a same building. When dealing with PON architectures (and more generally Point to Multipoint architectures), one easily understands that such a strategy might be drastically detrimental in terms of equipment costs, as the benefits of concentrators (here splitters) are lost for low demands. Therefore, considering PON architectures deployments, networks managers push for the following compromise: when the demand of a node exceeds a given value, called delocation threshold, then all its subscribers must be supplied by fibers of the latest ("latest" is to be defined according to the demand value) levels initiated by splitters located at the demand node.

Definition 2.1.1. *Let $\{e_0 = +\infty, e_1, ..., e_K\}$ be the set of delocation thesholds (positive integers sorted in decreasing order). Considering a demand node i, let k_i^{max} denote the maximum index k such as the threshold e_k exceeds the demand value d_i: $k_i^{max} = \max_{k=0,...,K}\{k \mid d_i < e_k\}$. Then the "Splitter Delocation" (SD) rule can be formulated as follows:*

$$\left\lceil \frac{d_i}{\prod_{j=k}^{K} m^j} \right\rceil \leq \mathbf{z}_i^k, \ \forall i \in V^*, \ \forall k = k_i^{max} + 1, ..., K \tag{2.5}$$

In other words, this rule means that if the demand at a node i exceeds a threshold $e_{k'}$ (with $k' > k_i^{max}$), it implies that:

- the demand also exceeds the given delocation thresholds for higher values of $k \geq k'$ if they exist (since threshold are in a decreasing order)
- the number of splitters of level $k \geq k'$ the rule will put onto the node i is large enough to *potentially* supply all the demand at node i

Remark 2.1.1. *If $e_K \geq \max_{i \in V} d_i$, then the SD rule never applies to the problem. This condition can also be written as $k_i^{max} = K$, $\forall i \in V$.*

For the sake of clarity, let us provide an example of how the splitter delocation rule can be applied (see Figure 2.1). There are 3 demand nodes on this instance, with respective demands 7, 10 and 7. For the sake of clarity, the deployed architecture is a single level splitters one. On the left, the optimal solution without the SD rule is presented. In order to quickly compute an optimal solution, let us assume for now that splitters are far more expensive than fibers and that it is always better to save a splitter than any number of fibers. On the right, the SD rule is applied with a delocation threshold e_1

set to 8, which means that splitters must be installed on the node with 10 demands. The optimal solution in this case is slightly different, due to the fact that fewer locating options are available for splitters, thus making the use of more fibers necessary compared to the solution when the SD rule is not applied.

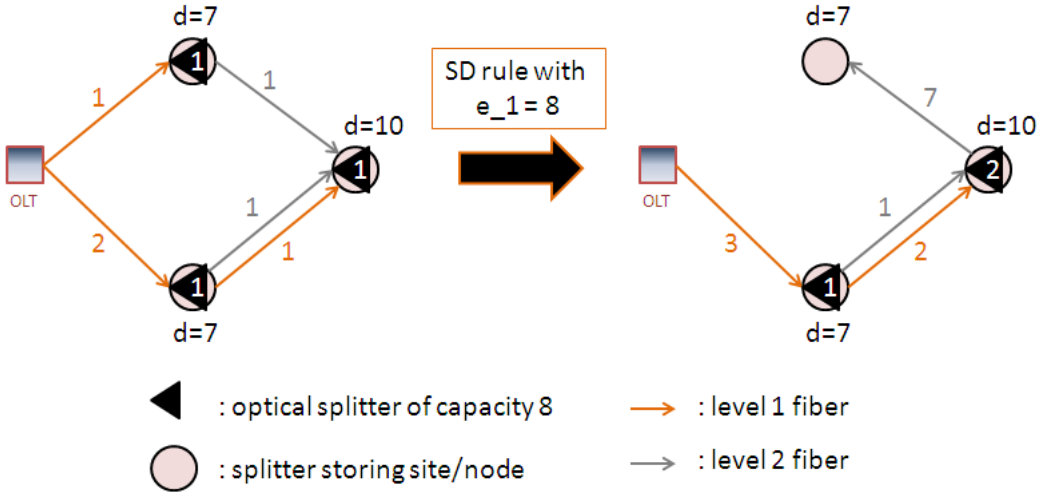

Figure 2.1: An illustration of the "Splitter Delocation" rule

However, with that single definition of the SD rule, there is no guarantee that clients at node i will be served by splitters located at node i if the rule applies on that node. That is what the following proposition is ensuring:

Proposition 2.1.1. *Considering the problem* (PON^K) *with constraint* (2.5)*, there exists an optimal solution such that, for each node* i *where* $k_i^{max} < K$ *(which means that the SD rule holds for that node), subscribers located on node* i *are all served by fibers from optical splitters of level* $k \geq k_i^{max}$ *also located on node* i*.*

Proof. The proof is only detailed for level K, as the same reasoning applies for all levels. Let i be a node of demand d_i such that $d_i \geq e_K$. The flow constraint (2.3) at this node is

$$\sum_{j \neq i} \left(\mathbf{f}_{ji}^{K+1} - \mathbf{f}_{ij}^{K+1}\right) + m^K \mathbf{z}_i^K = d_i + \mathbf{u}_i^{K+1}$$

If we denote by f_{ii}^{K+1} the number of clients at node i served by splitters located at that same node i. We will show that there is an optimal solution such that $f_{ii}^{K+1} = d_i$ if the SD rule is applied at node i. Since there is $\mathbf{z}_i^K \geq \left\lceil \frac{d_i}{m^K} \right\rceil$, it means that the number of level $K+1$ fibers generated at node i is greater than $m^K \left\lceil \frac{d_i}{m^K} \right\rceil \geq d_i$, which implies that $f_{ii}^{K+1} = d_i$ is a feasible configuration.

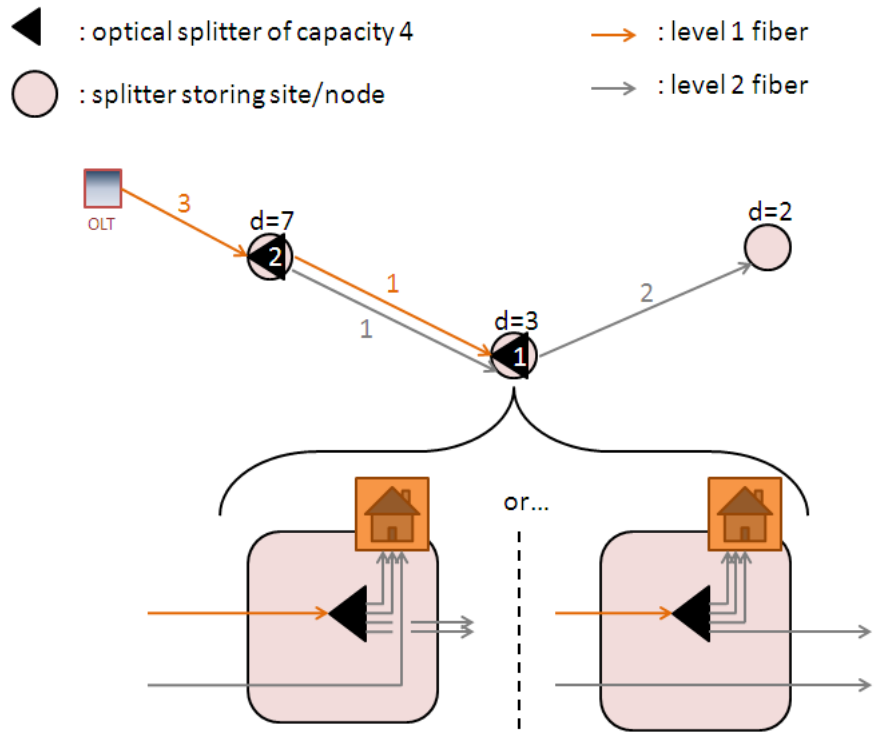

Figure 2.2: An illustration of local branching options under the SD rule with $e_1 = 2$

Figure 2.2 illustrates this principle. The optimal solution is represented for a single level PON architecture deployment with a delocation threshold equal to 2. Therefore, splitters of capacity 4 must be put in advance on the left-side and the central node. For the central node, one can easily see that there are 2 ways of connecting clients with level 2 fibers. The option on the right connects clients by using fibers coming only from the splitter located on that same node, and both options have the same optimal cost. Under the SD rule, one can easily be convinced that the "option on the right" will always be available. □

Corollary 2.1.1. *If delocation thresholds are such that $e_k = 0$ for all $k \geq k'$ with $k' \geq 1$, it means that the routing of level k fibers $(\forall k \geq k')$ is already done and does not need to be considered, therefore the problem (PON^K) and the problem $(PON^{k'-1}_{mod})$ are the same.*

The transformation to obtain $(PON_{mod}^{k'-1})$ is shown below, for the case of $k'=2$

$$
(PON_{mod}^{k'-1})\begin{cases}
\min\limits_{\mathbf{f},\mathbf{z},\mathbf{u}} & \sum\limits_{i=1}^{n}\sum\limits_{k=1}^{k'-1} C^k\mathbf{z}_i^k + \sum\limits_{(i,j)\in E}\sum\limits_{k=1}^{k'} c_{ij}^k\left(\mathbf{f}_{ij}^k+\mathbf{f}_{ji}^k\right) & \\
s.t. & \sum\limits_{j\neq i}\mathbf{f}_{ji}^1 = \mathbf{z}_i^1 + \sum\limits_{j\neq i}\mathbf{f}_{ij}^1,\ \forall i\in V^* & (2.6)\\
& \sum\limits_{j\neq i}\mathbf{f}_{ji}^{k'} + m^{k'-1}\mathbf{z}_i^{k'-1} = \left\lceil \frac{d_i}{\prod_{k=k'}^{K} m^k}\right\rceil + \sum\limits_{j\neq i}\mathbf{f}_{ij}^{k'} + \mathbf{u}_i^{k'},\ \forall i\in V & (2.7)\\
& \sum\limits_{k=1}^{K+1}\left(\mathbf{f}_{ij}^k+\mathbf{f}_{ji}^k\right)\leq b_{ij},\ \forall (i,j)\in E & (2.8)\\
& \mathbf{z}_i^k\in\mathbb{N},\ \ \forall i\in V^*; k=1,...,K; \mathbf{z}_0^k=0, \forall k=1,...,K & \\
& \mathbf{u}_i^k\in\mathbb{N},\ \ \forall i\in V^*; k=2,...,K+1; \mathbf{u}_0^k=0, \forall k=2,...,K & \\
& \mathbf{f}_{ij}^k\in\mathbb{N},\ \ \forall (i,j)\in E, k=1,...,K+1 &
\end{cases}
$$

This corollary implies that setting a delocation threshold to $e_k = 0$ implies that splitters (resp. fibers) variables of level greater than k (resp. $k+1$) can be set before solving the problem, thus reducing the number of required variables.

2.1.2 Modeling of the household grouping rule

PON deployment strategies that are purely driven by equipment CAPEX costs can lead to an important scattering of splitters amongst eligible storing sites. Be their CAPEX costs optimal, these deployment schemes should be avoided with regards to future maintenance costs (let us think of technicians rounds for instance). Indeed, the degree of splitter scattering will significantly impact the cost of technicians' rounds that will be needed for both preventive and curative maintenances. The more splitter sites are open, the longer maintenance rounds will be. Moreover, this rule tends to facilitate the network conception, as it will be more intelligible for network operators. Actually, databases describing the network are not always up-to-date and administrating a widely distributed network on a long-term run can be challenging without an efficient information system. As telecommunication operators are used to work with empirical knowledge, facilitating the acquisition of that knowledge through proposing a "simple", "intuitive" network is another key issue for OA&M concerns.

Therefore, considering PON deployments, network managers push for the following injunction: a site has the authorization for storing splitters of a given level only if the installed splitters can "deliver" fiber services for at least a given number of households (called household grouping threshold for this level). Another way of defining that injunction is to state that if a site is opened to a given level of splitters, then the number

of splitters installed on that node must be large enough to supply at least the grouping threshold in terms of households.

Definition 2.1.2. *Let us denote by HG^k the household grouping threshold for level k splitters ($\forall k = 1, ..., K$) and let $\mathbf{v}_i^k$ be a binary variable equal to 1 if and only if the $i \in V^*$ is open to level k splitters, and 0 otherwise. The OA&M "Household Grouping" (HG) rule can thus be formulated by means of the following set of logical constraints:*

$$\left(\mathbf{v}_i^k = 1\right) \Rightarrow \left(HG^k \leq \mathbf{z}_i^k \prod_{j=k}^{K} m^j \right) \quad \forall k = 1..K, \forall i \in V^* \tag{2.9}$$

$$\left(\mathbf{v}_i^k = 0\right) \Rightarrow \left(\mathbf{z}_i^k = 0\right) \quad \forall k = 1..K, \forall i \in V^* \tag{2.10}$$

$$\mathbf{u}_i^{k+1} \leq \left(m^k - 1\right) \mathbf{v}_i^k \quad \forall k = 1..K, \forall i \in V^* \tag{2.11}$$

Note that constraints (2.11) were proven automatically satisfied in (PON^K) by authors in [21] (see remark 1.1.3 in section 1.1.2). They must be added here since constraints (2.9) and (2.10) are imposing a minimal number of splitters to be put on every node, and there is no guarantee anymore that the model will not add optical splitters to satisfy those constraints that are, in fact, useless.

Also note that HG^k thresholds are not the exact number of clients that must be served from a node opened to level k splitters. Actually, HG^k are the maximal number of clients such that $\left\lceil \frac{HG^k}{\prod_{j=k}^{K} m^j} \right\rceil$ splitters will be able to serve. Indeed, for each level of splitter at a given node, there can be at most $m^k - 1$ unused fibers going out of these splitters. And by the influence of successive splitter layers, one can not control the effective number of clients connected to splitters installed at a node. Therefore, HG^k thresholds must be interpreted as a number designed to impose a minimal number of splitters per node than a real number of connected households.

Figure 2.3 gives an example of how this rule is applied. Without applying this rule (left part of the figure), splitters can be installed in any eligible site, without considering the total number of sites to be open, and especially without considering how many customers are supplied by a storing site. On the contrary, let us set the customer threshold to 24 to impose the grouping of splitters in storing sites. Then, in this example (right part of the figure), all splitters should be installed at a single storing site, leaving others empty, while optimizing the fiber cost. Of course, adding this constraint will impose more fibers reaching their target through longer paths, which is detrimental in terms of cost.

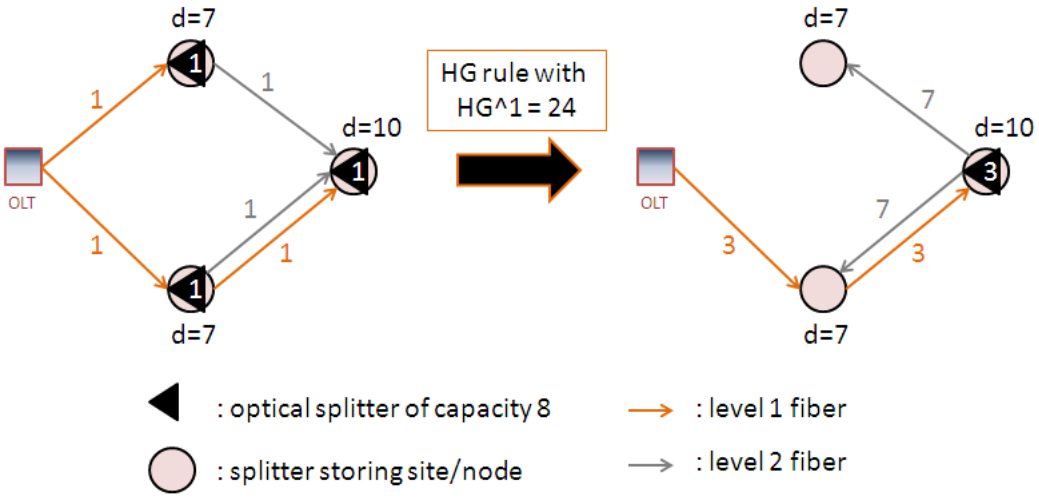

Figure 2.3: An illustration of the "Household Grouping" rule

2.2 Theoretical analysis of OA&M constraints

In the following, we will refer to the problem differently according to the rules that are taken into account. The (PON^K) problem will thus be denoted by:

- (PON^K_{SD}) if it only includes the SD rule (i.e. constraints (2.5))
- (PON^K_{HG}) if it only includes the HG rule (i.e. constraints (2.9),(2.10) and (2.11))
- $(PON^K_{OA\&M})$ if it includes both rules (and related constraints)

In order to give an insight on the overcosts induced by these new rules and constraints on the problem, we present some pathological cases where the potential detrimental effect of these rules will be shown. We will expose theses pathological cases for a single level PON architecture for the sake of clarity, but the phenomena that are exposed in this section can be generalized to K-levels PON architectures. Furthermore, we also provide some feasibility conditions for (PON^K_{HG}) and (PON^K_{OAM}).

2.2.1 A pathological case for the optical splitter delocation rule

Let us denote by $SM^{e_1}_{m^1}$ the smallest multiple of m^1 greater than the level 1 delocation threshold e_1. We consider a star-graph with m^1 demand nodes of $SM^{e_1}_{m^1}+1$ households, each of them directly connected the OLT (assumed to be the center of the star-graph) through a link of sufficient capacity (at least greater than $SM^{e_1}_{m^1}+1$) and with a unitary routing cost of c, whatever the fiber level used. An example of how this star-graph is built is given in figure 2.4 for $m^1 = 8$ and $e_1 = 50$ (which gives $SM^{e_1}_{m^1} = 56$). In this case,

we authorize splitters to be placed at the OLT (which can be seen as a double node, the OLT and a normal one, with a link between those two of length equal to zero).

In such context, one should easily be convinced that:

- The optimal solution of PON^1 consists in
  - installing $\frac{SM_{m^1}^{e_1}}{m^1}$ level 1 splitters on each demand node and 1 splitter at the OLT
  - routing $\frac{SM_{m^1}^{e_1}}{m^1}$ level 1 and one level 2 fibers from the OLT to each demand node

  which leads to an optimal deployment cost of $SM_{m^1}^{e_1}(C^1 + c) + C^1 + m^1 c$.

- The optimal solution of PON_{SD}^1 consists in
  - installing $\left(\frac{SM_{m^1}^{e_1}}{m^1} + 1\right)$ level 1 splitters at each demand node
  - routing $\left(\frac{SM_{m^1}^{e_1}}{m^1} + 1\right)$ level 1 fibers from the OLT to each demand node

  which leads to an optimal deployment cost of $(SM_{m^1}^{e_1} + m^1)(C^1 + c)$.

Then the CAPEX overcosts induced by the SD rule are $(m^1 - 1)C^1$, which can prove to be important due to the relatively high cost of splitters equipments (compared to fibers). More generally, this rule reduces the fiber aggregation capacity of optical splitters, which imposes us to place splitters that may not be used at their full capacity.

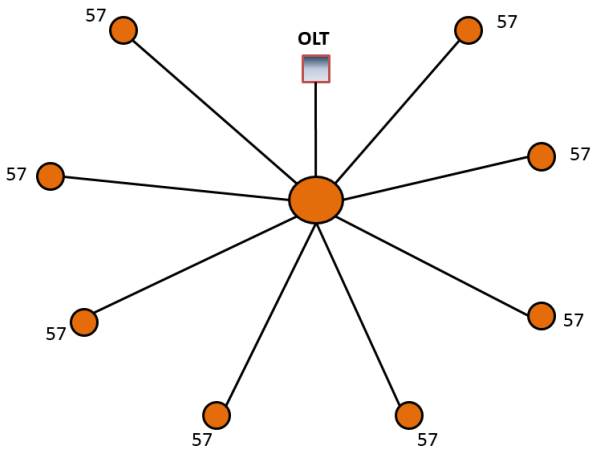

Figure 2.4: An example of the pathological case for the SD rule with $m^1 = 8$ and $e_1 = 50$

2.2.2 A pathological case for the household grouping rule

Let us consider a star-graph with $\left\lceil \frac{HG^1}{m^1} \right\rceil$ demand nodes of m^1 households, each of them being directly connected to the OLT (assumed to be at the center of the star-graph) through a link of sufficient capacity (at least greater than m^1) and with a unitary routing cost of c whatever the fiber level used. An example of how this star-graph is built is given in figure 2.5 for $m^1 = 8$ and $HG^1 = 75$ (which gives a star-graph with 10 demand nodes). Like the example given in section 2.2.1, we authorize splitters to be placed on the OLT. In such context, one should easily be convinced that:

- The optimal solution of PON^1 consists in
  - installing 1 level 1 splitters on each demand node
  - routing 1 level 1 fiber from the OLT to each demand node

  which leads to an optimal deployment cost of $\left\lceil \frac{HG^1}{m^1} \right\rceil \left(C^1 + c\right)$.
- The optimal solution of PON^1_{HG} consists in
  - installing $\left\lceil \frac{HG^1}{m^1} \right\rceil$ level 1 splitters at the OLT
  - routing m^1 level 2 fibers from the OLT to each demand node

  which leads to an optimal deployment cost of $\left\lceil \frac{HG^1}{m^1} \right\rceil \left(C^1 + m^1 c\right)$.

Then the CAPEX overcosts induced by the HG rule are $\left\lceil \frac{HG^1}{m^1} \right\rceil \left(m^1 - 1\right) c$, which is potentially unbounded (thinking of c being arbitrarily large). More generally, with high values of HG^k, this will impose a very small number of splitters sites to be open, which will lead to a higher fiber cost because there will be fewer routing options.

2.2.3 Feasibility conditions and thresholds' effective ranges for the problem with OA&M constraints

In this section, we examine the feasibility of OA&M related problems, compared to PON^K. Therefore, we will always work under the hypothesis that PON^K has at least one feasible solution. We also give conditions under which OA&M rules are not effective.

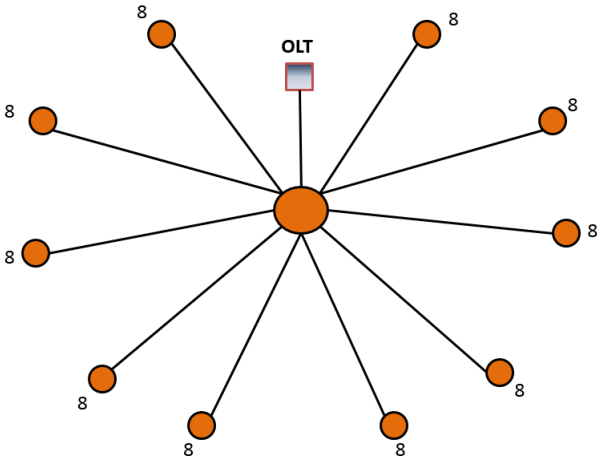

Figure 2.5: An example of the pathological case for the HG rule with $m^1 = 8$ and $HG^1 = 75$

Feasibility of PON^K_{SD}

First, we note the following proposition:

Proposition 2.2.1. *Let us consider a given instance of PON^K and a given set of splitter delocation thresholds $\{e_k | k = 1...K\}$. If $e_K \geq \max_{i \in V} d_i$, then the SD rule is never applied on any node of the graph. In other words, we have:*

$$v(PON^K) = v(PON^K_{SD})$$

Proof. It is immediate that if every delocation threshold is greater than the demand in any node of the graph, then the rule won't apply. Moreover, by definition 2.1.1, delocation thresholds are sorted in decreasing order, so that the K-th one is the smallest. Therefore, one only needs to consider the last level threshold to compare it with the whole set of demands. □

Then, we can give the following property on the SD rule's feasibility:

Remark 2.2.1. *If PON^K_{SD} is feasible for any set of delocation thresholds, then PON^K is feasible. However, if PON^K is feasible, it does not imply that PON^K_{SD} is feasible for any set of delocation threshold.*

Proof. The implication is straightforward since any valid solution of PON^K_{SD} is also a valid solution of PON^K. On the contrary, figure 2.6 gives a counterexample that shows a graph with a feasible solution of PON^1, and an infeasibility in PON^1_{SD} when the delocation threshold is set to $e_1 = 1$.

□

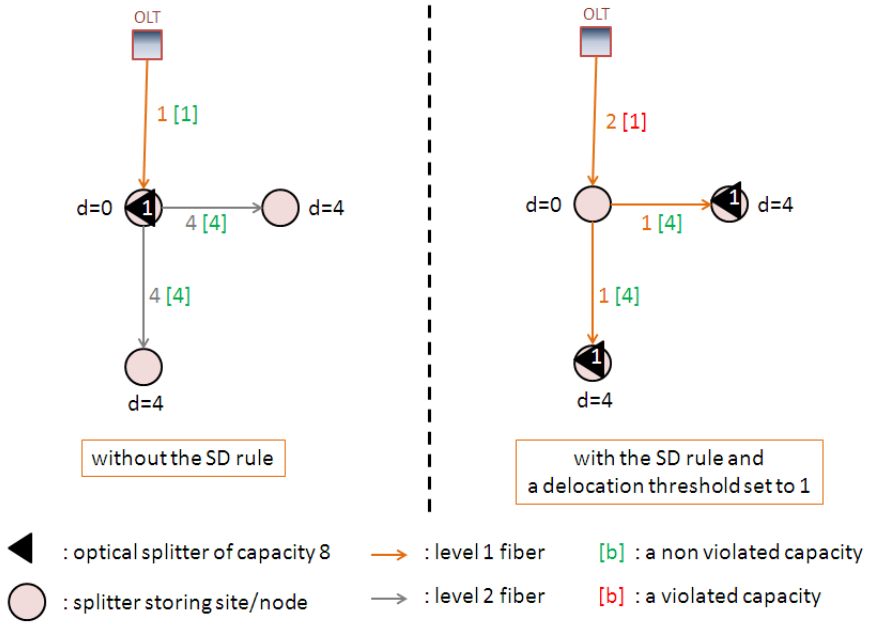

Figure 2.6: A counterexample of a graph where the SD rule can make the problem infeasible with $e_1 = 1$

Lemma 1. *If PON^k_{SD} is feasible when all its delocation thresholds are equal to 0, then it is feasible for any set of delocation thresholds.*

Proof. A solution of PON^k_{SD} where all delocation thresholds are set to 0 reduces to installing splitters to every demand node, thus only using level 1 fibers in the graph. One can easily see that a solution that respects the SD rule with thresholds set to 0 also respects the SD rule with thresholds of higher values. □

On the basis of these observations, we can derive the following sufficient feasibility condition for the SD rule:

Proposition 2.2.2. *A sufficient feasibility condition for PON^K_{SD}, for every set of delocation threshold possible, is:*

$$b_{ij} \geq \sum_{i \in V} \left\lceil \frac{d_i}{\prod_{j=1}^{K} m^j} \right\rceil, \ \forall (i,j) \in E$$

Proof. By Lemma 1, a condition that ensures the feasibility of PON^k_{SD} for delocation thresholds that are set to 0 is also valid for every set of thresholds. As a solution of PON^k_{SD} with thresholds sets uses $\sum_{i \in V} \left\lceil \frac{d_i}{\prod_{j=1}^{K} m^j} \right\rceil$ level 1 fibers at the same time, having the capacities of every edge superior or equal to that number is a sufficient feasibility condition for PON^k_{SD}. □

Feasibility of PON^K_{HG}

As for the SD rule, let us examine when the HG rule is not effective.

Proposition 2.2.3. *Let us consider a given instance of PON^K and a given set of household grouping thresholds $\{HG^k|k=1...K\}$. If $HG^k \leq \prod_{j=k}^{K} m^j$, then the HG rule is never applied for level k splitters. If that inequality holds for all $k=1...K$, then the rule is never applied at all. In other words, we would have:*

$$v(PON^K) = v(PON^K_{HG})$$

Proof. Let us recall constraint (2.9) that imposes the minimal number of level k splitters one should install on a splitter site opened for level k:

$$\left(\mathbf{v}_i^k = 1\right) \Rightarrow \left(HG^k \leq \mathbf{z}_i^k \prod_{j=k}^{K} m^j\right) \quad \forall k = 1..K, \forall i \in V^* \quad (2.9)$$

That constraint will be ineffective if it only imposes that $\mathbf{z}_i^k \geq 1$, which happens if $HG^k \leq \prod_{j=k}^{K} m^j$. □

However, unlike the SD rule, the HG rule does not always guarantee feasibility. Indeed, it is rather intuitive that setting a threshold $HG^k \geq \sum_{i \in V} d_i$ is not really relevant and could lead to infeasibility. Another source of infeasibility for the HG rule is arc capacities. First, let us give the following valid inequality for PON^K_{HG} (and even $PON^K_{OA\&M}$).

Proposition 2.2.4. *Let us denote the minimal useful fiber flow of a level k splitter storing site under the HG rule by:*

$$\delta^k = m^k \left\lceil \frac{HG^k}{\prod_{j=k}^{K} m^j} \right\rceil - \left(m^k - 1\right), \quad \forall k = 1...K$$

and the maximal number of level k splitter storing sites one can open in order to supply the whole demand $\overline{N^k_{site}}$ for which we give the following recursive definition:

$$\overline{N^K_{site}} = \left\lfloor \frac{\sum_{i \in V} d_i}{\delta^K} \right\rfloor$$

$$\overline{N^k_{site}} = \left\lfloor \frac{\overline{N^{k+1}_{site}} \left\lceil \frac{HG^{k+1}}{\prod_{j=k+1}^{K} m^j} \right\rceil}{\delta^k} \right\rfloor, \quad \forall k = 1...K-1 \text{ if } K \geq 2$$

Then

$$\sum_{i \in V} \mathbf{v}_i^k \leq \overline{N_{site}^k} \tag{2.12}$$

is a valid inequality for PON_{HG}^K *and* $PON_{OA\&M}^K$.

Proof. For any splitter level k, one can compute the minimal number of subscribers that can be served by installed splitters at a given node under the HG rule. The minimal fiber output of p level k splitters is $m^k p - (m^k - 1)$ as every installed splitter must be used to its full capacity, except only one which supplies at least one fiber. Denoting by δ^k this minimal output for a splitter site when the HG rule is applied, it thus comes:

$$\delta^k = m^k \left\lceil \frac{HG^k}{\prod_{j=k}^K m^j} \right\rceil - \left(m^k - 1\right), \ \forall k = 1...K$$

Let us denote by $\overline{N_{site}^k}$ the maximum number of level k splitter sites to open in order to supply the whole demand, that is, considering that every opened splitter site will only supply the minimal amount of demand d^k. For the last splitter level, we obviously have

$$\overline{N_{site}^K} = \left\lfloor \frac{\sum_{i \in V} d_i}{\delta^K} \right\rfloor$$

Let us recursively compute $\overline{N_{site}^k}$ from $\overline{N_{site}^{k+1}}$, for all $k = 1...(K-1)$. Let us notice that an upper bound for the number of level k splitters to be installed is given by $\overline{N_{site}^k} \left\lceil \frac{HG^k}{\prod_{j=k}^K m^j} \right\rceil$. As previously defined, a lower bound for the amount of fibers a level k splitter site can supply is given by δ^k. Therefore, the maximum number of level k splitter sites one can open is given by $\overline{N_{site}^k} = \left\lfloor \frac{\overline{N_{site}^{k+1}} \left\lceil \frac{HG^{k+1}}{\prod_{j=k+1}^K m^j} \right\rceil}{\delta^k} \right\rfloor$, $\forall k = 1...K-1$ if $K \geq 2$. Then, the total number of opened level k splitter sites cannot exceed $\overline{N_{site}^k}$. This inequality holds for PON_{HG}^K and the addition of the SD rule does not change it so it also holds for $PON_{OA\&M}^K$. □

We thus give the following necessary conditions on PON_{HG}^K feasibility:

Proposition 2.2.5. *Necessary feasibility conditions on* HG^k *thresholds values* ($\forall k =$

1...K) for PON^K_{HG} *are*

$$HG^K \leq \left\lceil \frac{\sum_{i\in V} d_i}{m^K} \right\rceil m^K \tag{2.13}$$

$$HG^k \leq \left\lceil \frac{\overline{N^{k+1}_{site}} \left\lceil \frac{HG^{k+1}}{\prod_{j=k+1}^K m^j} \right\rceil}{m^k} \right\rceil \prod_{j'=k}^K m^{j'}, \quad \forall k = 1...K-1 \tag{2.14}$$

Proof. Let us consider the last splitter level K. We are trying to determine what is the highest value possible for HG^K so that the problem remains feasible. One can easily see that the higher the value of HG^K, the fewer the number of opened splitter sites. Therefore, the highest value must be coinciding with a single opened splitter site. For j being that only opened node, the number of installed level K splitter on that node must be such that:

$$\mathbf{z}^K_j \leq \left\lceil \frac{\sum_{i\in V} d_i}{m^K} \right\rceil$$

Since $\mathbf{z}^k_j \geq \left\lceil \frac{HG^K}{m^K} \right\rceil$, HG^K threshold must verify:

$$\frac{HG^K}{m^K} \leq \left\lceil \frac{HG^K}{m^K} \right\rceil \leq \left\lceil \frac{\sum_{i\in V} d_i}{m^K} \right\rceil$$

which leads to the final result, that is the feasibility condition for level K splitters. For lower levels, we apply the same reasoning, but one needs to take next level splitters instead of the demand. With $\overline{N^k_{site}}$ defined as in Proposition 2.2.4, one can easily see that the maximum number of next level splitters that may be installed equals $\overline{N^{k+1}_{site}} \left\lceil \frac{HG^{k+1}}{\prod_{j=k+1}^K m^j} \right\rceil$. □

Of course, this condition is only necessary since capacity constraints can also lead to infeasibility. Indeed, imposing a large number of splitters to be put per site also imposes a potentially large number of fibers coming out of a single node, which can lead to infeasibility.

Feasibility of $PON^K_{OA\&M}$

Applying simultaneously the SD rule and the HG rule can also imply problem infeasibility. Indeed, since the SD rule will tend to open splitter sites if they meet the requirement, the HG rule has an opposite objective since it tends to limit the total number of opened splitter sites.

Proposition 2.2.6. *We have the following necessary feasibility condition for the* $PON^K_{OA\&M}$ *problem:*

$$\sum_{i \in V} \chi_{\{d_i \geq e_k\}} \leq \overline{N^k_{site}}, \ \forall k = 1...K \tag{2.15}$$

where χ *stands for the indicator function.*

Proof. This result is straight-forward derived from Proposition 2.2.4, noticing that $\sum_{i \in V} \chi_{\{d_i \geq e_k\}}$ is a lower bound of the number of level k splitter sites. □

2.3 Solving the problem with OA&M constraints

The method used to solve the problem is a mere branch-and-bound approach, based on the CPLEX framework. However, the model $PON^K_{OA\&M}$ as it is now does not fit the general integer linear program scheme.

2.3.1 Linearizing the household grouping rule constraints

Linearizing constraints (2.9) and (2.10) can be made via the use of a so-called big M formulation.

Proposition 2.3.1. *Constraints* (2.9) *and* (2.10) *can be linearised as follows:*

$$\mathbf{z}^k_i \leq M^k_i \mathbf{v}^k_i \ \ \forall k = 1...K, \forall i \in V \ \textit{(with } M^k_i \textit{ a large constant)} \tag{2.16}$$

$$\left\lceil \frac{HG^k}{\prod_{j=k}^K m^j} \right\rceil \mathbf{v}^k_i \leq \mathbf{z}^k_i \ \ \forall k = 1...K, \forall i \in V \tag{2.17}$$

However, big M formulations often lead to poor continuous relaxations. It is thus essential to provide the lowest value possible, yet valid, for M^k_i. We provide two different calculations, based on the total demand and the capacity of incoming edges of a node.

Proposition 2.3.2.

$$\alpha^k_i = \begin{cases} \left\lceil \frac{\sum_{i \in V} d_i}{m^K} \right\rceil & \forall i \in V, k = K \\ \left\lceil \frac{\overline{N^{k+1}_{site}} \left\lceil \frac{HG^{k+1}}{\prod_{j=k+1}^K m^j} \right\rceil}{m^k} \right\rceil & \forall i \in V, \forall k = 1...K-1 \end{cases}$$

is valid for equation (2.16)*.*

Proof. This result is derived from Proposition 2.2.5. Indeed, we already know what would be the maximal number of splitters one could install on a single node per level, therefore it is a suitable value for M_i^k. □

However, depending on values of $HG^k, k = 1...K$, α_i^k values may still be high. And as we noticed before, capacity constraints are another limitation for the number of installable splitters at a node.

Proposition 2.3.3.

$$\beta_i^k = \begin{cases} \max\limits_{M\in\mathbb{N}} \left\{ M : M + \left(m^1(M-1) + 1 - \left\lceil \frac{d_i}{\prod_{j=1}^{K} m^j} \right\rceil \right)^+ \leq \sum\limits_{j:(i,j)\in E} b_{ji} \right\} & \forall i \in V, k = 1 \\ \max\limits_{M\in\mathbb{N}} \left\{ M : \left\lceil \frac{M}{\prod_{j=1}^{k-1} m^j} \right\rceil + \left(m^k(M-1) + 1 - \left\lceil \frac{d_i}{\prod_{j=k}^{K} m^j} \right\rceil \right)^+ \leq \sum\limits_{j:(i,j)\in E} b_{ji} \right\} & \forall i \in V, \forall k > 1 \end{cases}$$

is valid for equation (2.16).

Proof. Let M be the number of level k splitters installed at site $i \in V$, and d_i the demand of that site (which can be equal to zero if there is no demand on the node). Let us compute the minimum cumulative number of incoming and outgoing fibers due to these splitters.

If $k = 1$, the number of incoming level 1 fibers supplying our M number of level 1 splitters is equal to M. If $k > 1$, then that number of incoming level 1 fibers is equal to $\left\lceil \frac{M}{\prod_{j=1}^{k-1} m^j} \right\rceil$ which is by definition the least capacity consuming way of supplying level k splitters on a node. Moreover, by constraints (2.11), the minimum of level $k+1$ fibers produced by these splitters is $m^k(M-1)+1$. At most $\left\lceil \frac{d_i}{\prod_{j=k}^{K} m^j} \right\rceil$ of these fibers are internally used to supply the demand at node i (by the same principle showed in proof of Proposition 2.1.1). Therefore, a lower bound for the number of produced level $k+1$ fibers that are not used to supply the node demand is $\left(m^k(M-1) + 1 - \left\lceil \frac{d_i}{\prod_{j=k}^{K} m^j} \right\rceil \right)^+$. Noting that one level $k+1$ fiber produced which is not used to supply the internal demand will at least induce one outgoing level $k+2$ fiber (exactly 1 fiber of level $k+1$ if going directly out of node i, at least 1 level $k+2$ fiber if going through only a level $k+1$ splitter, ...), we conclude on the minimum number of incoming and outgoing fibers due to M splitters of level k, which has to be lower to the cumulative capacity of the incoming/outgoing edges of the node i. □

Remark 2.3.1. *The function* $\left\lceil \frac{M}{\prod_{j=1}^{k-1} m^j} \right\rceil + \left(m^k(M-1) + 1 - \left\lceil \frac{d_i}{\prod_{j=k}^{K} m^j} \right\rceil \right)^+$ *is an increasing function in* M*. Therefore, finding the maximal value of* M *such that conditions exposed in Proposition 2.3.3 are respected is an easy problem in terms of complexity (it can be written as a 2 integer linear program variable).*

Corollary 2.3.1.

$$M_i^k = \min\left\{\alpha_i^k, \beta_i^k\right\}, \quad \forall i \in V, \forall k = 1...K$$

is valid for equation (2.16).

2.3.2 Development of a pre-processing routine for the household grouping rule

From results of Corollary 2.3.1, we can derive a preprocessing scheme, considering that:

- Corollary 2.3.1 (and more generally any set of valid M_i^k values) provides us with an upper bound of the number of splitters at a given storing site
- The Household Grouping rule implies a minimum number of splitters to be installed if a site is open

From these considerations, some **v** and **z** variables can be preprocessed, as shown in Algorithm 2.

Algorithm 2 HG rule preprocessing algorithm for level k splitters

Require: V, HG^k, M_i^k, $\forall i \in V$

1: **for** $i = 1, ..., |V|$ **do**
2: **if** $M_i^k < \left\lceil \frac{HG^k}{\prod_{j=k}^{K} m^j} \right\rceil$ **then**
3: $\mathbf{z}_i^k \leftarrow 0$
4: $\mathbf{v}_i^k \leftarrow 0$
5: **end if**
6: **end for**

2.3.3 Empirical analysis of the influence of OA&M constraints on CAPEX costs

Tests are performed on a 10 real-life instances basis. Theses are selected so as to be representative of two types of areas where FTTH PON deployment is impending: first, local areas with very high density of population (Net_1 - Net_7) and second, local areas of moderate density of population (Net_8 - Net_{14}). Those instances are extracted from Orange's databases, over local areas in french cities of various sizes. Table 2.1 gives for each instance the number of nodes and edges, the total demand, the mean demand and the associated standard deviation (σ), as for capacities of the graph. If capacities are very similar over all these instances, we can note that very high density instances have a lot

Instances			Demands				Capacities	
Name	$\|V\|$	$\|E\|$	$\sum_i d_i$	Mean	σ	$\max_i d_i$	Mean	σ
Net_1	808	1101	46293	57.36	67.38	423	1198.17	1090.21
Net_2	52	65	1828	35.84	52.64	199	683.51	726.77
Net_3	235	313	12604	53.86	60.48	300	1172.32	1211.21
Net_4	452	613	25420	56.36	68.18	423	1151.77	1118.87
Net_5	322	421	16068	50.06	64.70	423	1163.86	1105.95
Net_6	392	537	22326	57.10	65.00	343	1130.35	1096.31
Net_7	229	297	11946	52.39	62.83	342	1188.24	1189.07
Net_8	1624	1987	23734	14.62	24.63	252	1319.81	1656.52
Net_9	602	730	8778	14.61	29.21	252	1373.52	1539.59
Net_{10}	1258	1544	18149	14.44	22.46	222	1285.51	1580.21
Net_{11}	234	297	2345	10.06	21.25	166	1049.30	872.62
Net_{12}	955	1173	14000	14.68	20.84	144	1269.91	1441.58
Net_{13}	449	562	5652	12.62	22.07	192	1355.27	1253.19
Net_{14}	117	160	1147	9.89	20.92	115	788.14	437.16

Table 2.1: Instances statistical description

more demand to be served for fewer nodes than moderate density ones. Another feature that is not displayed in table 2.1 is that very high density instances' graphs are in the shape of a meshing, while moderate density instances' graphs are much more dispatched, offering fewer paths from one point to another, which means fewer optimizing options. Moreover, moderate density areas are wider than high density ones, which implies longer edges over the instance. All this emphasizes the fact that the splitter cost of a solution will be, in proportion, higher for high density areas than moderate ones, and vice versa.

We now solve the problem PON^K (which means without taking any OA&M rules into account) over these instances with 3 different architectures in order to make comparisons with results of $PON^K_{OA\&M}$ problems. Let us define the 3 PON architectures for a total splitting ratio of 64 (splitters capacities were chosen accordingly to what is done on the field):

- Architecture 1: 1 splitter level with the following set of capacities
  - $m^1 = 64$
- Architecture 2: 2 splitter levels with the following set of capacities

	m^k					
	2	4	8	16	32	64
C^k	69	92	138	230	414	782

Table 2.2: Splitter costs

  - $m^1 = 2$
  - $m^2 = 32$

- Architecture 3: 3 splitter levels with the following set of capacities
  - $m^1 = 2$
  - $m^2 = 8$
  - $m^3 = 4$

For each instance, we solve the problem PON^K with each architecture. Splitter costs are set according to table 2.2. The cost of a fiber is the same for every level and is set to 0.1 per meter. This cost was computed as an average cost considering various fiber cables. Even though these splitter and fiber costs may not be strictly exacts, they are right in their order of magnitude.

Instances were solved by CPLEX 12.2 over a maximum time of 500 seconds (with an additional polishing time of 50 seconds, which is a local search procedure). For each architecture, we indirectly give the optimal cost by providing the optimal splitter cost $C\mathbf{z}$ and the optimal fiber cost $c\mathbf{f}$ (that have to be summed). We also give the best gap to the continuous relaxation of the problem we got if the optimal solution was not found within the time limit. We also give the CPU time needed/allowed for each instance.

Table 2.3 presents our results. The first observation is that very few instances are solved to optimality. Indeed, only Net_2 and Net_{14} are, since they are the smallest ones, and only for architectures 1 and 2. Nevertheless, gaps to optimality are also very small. One can notice that moderate density instances seems to be harder to solve, which was to be expected. Another interesting feature is that the average ratio $\frac{C\mathbf{z}}{C\mathbf{z} + c\mathbf{f}}$ is higher for high density areas (90%) than for moderate density ones (80%), as it was also expected. This is indeed due to the smaller number of routing options we have in moderate density areas and the fact that paths are longer.

The cost of the optical splitter delocation rule

The splitter delocation rule restricts possibilities of splitter placement. This will have a detrimental impact on the cost of the solution. Let us show how that cost increases as

the delocation rule is getting strengthened. We pick 2 small instances out of our set. We choose small ones mainly for practical reasons. Since they are small, they are also easier to solve, even for a 3 level PON architecture (see table 2.3). Net_2 and Net_{14} are picked in order to have one instance of both density profiles. The 3^{rd} architecture defined in section 2.3.3 is chosen. Many sets of delocation thresholds are possible for these instances. Let us describe our 4 sets of thresholds:

- $\left\{e_0^1 = +\infty,\ e_1^1 = \gamma_{SD},\ e_2^1 = \gamma_{SD},\ e_3^1 = \gamma_{SD}\right\}$ for $\gamma_{SD} = 1, ..., \max_{i \in V} d_i$
- $\left\{e_0^2 = +\infty,\ e_1^2 = 2\gamma_{SD},\ e_2^2 = \gamma_{SD},\ e_3^2 = \gamma_{SD}\right\}$ for $\gamma_{SD} = 1, ..., \max_{i \in V} d_i$
- $\left\{e_0^3 = +\infty,\ e_1^3 = 2\gamma_{SD},\ e_2^3 = 2\gamma_{SD},\ e_3^3 = \gamma_{SD}\right\}$ for $\gamma_{SD} = 1, ..., \max_{i \in V} d_i$
- $\left\{e_0^4 = +\infty,\ e_1^4 = 3\gamma_{SD},\ e_2^4 = 2\gamma_{SD},\ e_3^4 = \gamma_{SD}\right\}$ for $\gamma_{SD} = 1, ..., \max_{i \in V} d_i$

Of course, many other sets of parameters are possible, but those 4 are enough to emphasize the general behaviour of the SD rule with regard to the impact on the solution cost. For each instance, we are going to solve the problem PON^3_{SD} with every possible threshold set as defined before, that is for every possible value of γ_{SD} in the instance (which means that Net_2 will be solved 796 times, and Net_{14} will be solved 460 times). The bound on γ_{SD} value is set thanks to Proposition 2.2.1.

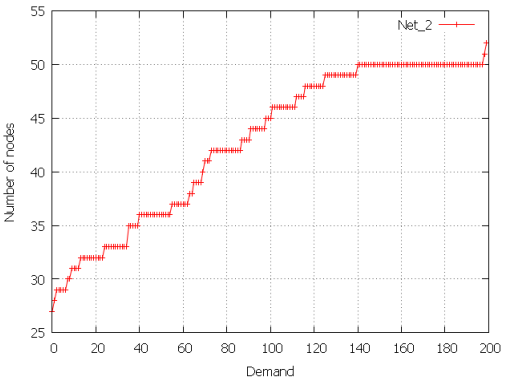

Figure 2.7: Demand repartition for Net_2

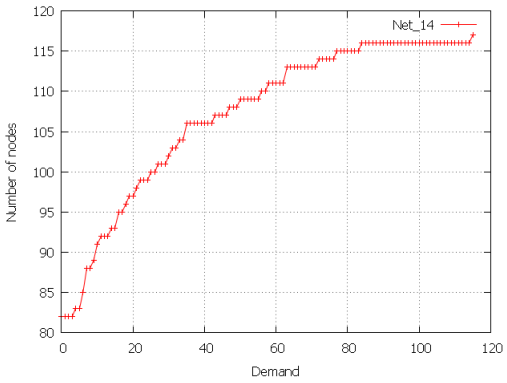

Figure 2.8: Demand repartition for Net_{14}

Finally, before solving our problems, let us give the demand profile of each instance (see Figure 2.7 and 2.8 that give, for a given demand, the number of nodes in the graph that have a demand that is less or equal). As we can see, Net_2 has 51.9% of its nodes with a demand that equals 0. These nodes are mostly network nodes where a splitter room may be open but that is not directly linked to a customer. That percentage goes up to 70.1% for Net_{14}. The demand repartition is not uniform, but curves on figures 2.7 and 2.8 are almost continuous.

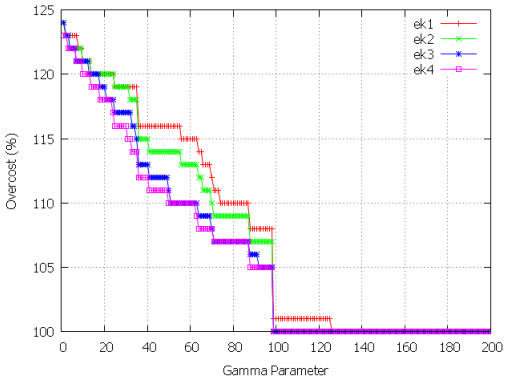

Figure 2.9: Overcosts due to the SD rule in function of parameter γ_{SD} for Net_2

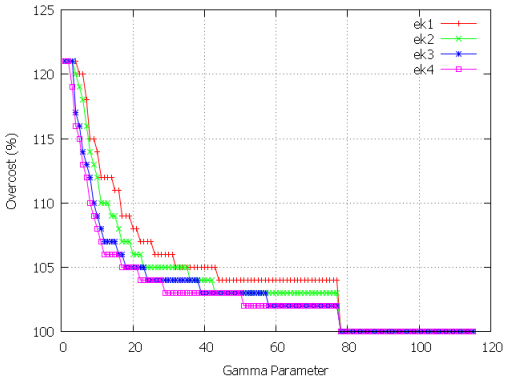

Figure 2.10: Overcosts due to the SD rule in function of parameter γ_{SD} for Net_14

Figures 2.9 and 2.10 show results for these tests. As one can see, the overcost (which is given as a percentage of the best value found for PON^K) decreases as γ_{SD} increases, which was to be expected since higher values of it means a less constrained SD rule. There are two things that need to be noticed. First, in both cases, the decreasing of the overcost is sharp for low values of γ_{SD} in the moderate density area, almost linear for the high density area. After the inflexion point, the progress is much slower. This must be taken into account while setting our delocation thresholds. Indeed, unless the future OA&M gain is important enough to justify the CAPEX overcost, the SD rule shall not be applied to every node for every splitter level. Another thing that must be noted is that applying the SD rule differently according to the splitter level has an interest. Indeed, the overcost gap between e_k^1 and e_k^4 thresholds can exceed 5% for the same γ_{SD} value. The second important thing is the sudden overcost drop for high values of γ_{SD} (for $\gamma_{SD} = 99$ in Net_2, $\gamma_{SD} = 78$ in Net_{14}). This drop cannot be explained by the demand repartition given in figures 2.7 and 2.8 since there is not a specifically high number of nodes with those precise demands. The explanation must be more subtle. Indeed, one can see that the overcost goes down step by step. It is just the "last" step that is bigger than the other ones. So far, we can only conjecture that when γ_{SD} increases, there will be a point where it will free a small set of essential nodes from the SD rule, thus enabling much more good solutions to be found. We know that these nodes have a specific demand so we can identify them, but their "property" is probably not linked to that particular amount of demand since it must have something to do with the local graph topology around this node.

The cost of the household grouping rule

The household grouping rule consists in forcing splitters to be located in a limited number of nodes. Thus, it restrict the number of fiber routing solutions, which will increase the solution cost. As for the SD rule, let us show how the cost increases as the HG rule is strengthened. We perform our tests on the same basis as for the SD rule. We define 4 possible sets of Household Grouping thresholds:

- $\left\{HG_1^1 = 10\gamma_{HG},\ HG_1^2 = 10\gamma_{HG},\ HG_1^3 = 10\gamma_{HG}\right\}$ for $\gamma_{HG} \in \mathbb{N}^+$
- $\left\{HG_2^1 = 20\gamma_{HG},\ HG_2^2 = 20\gamma_{HG},\ HG_2^3 = 10\gamma_{HG}\right\}$ for $\gamma_{HG} \in \mathbb{N}^+$
- $\left\{HG_3^1 = 50\gamma_{HG},\ HG_3^2 = 30\gamma_{HG},\ HG_3^3 = 10\gamma_{HG}\right\}$ for $\gamma_{HG} \in \mathbb{N}^+$
- $\left\{HG_4^1 = 30\gamma_{HG},\ HG_4^2 = 50\gamma_{HG},\ HG_4^3 = 10\gamma_{HG}\right\}$ for $\gamma_{HG} \in \mathbb{N}^+$

We solve PON_{HG}^3 for every set of parameters, for every valid value of γ_{HG}.

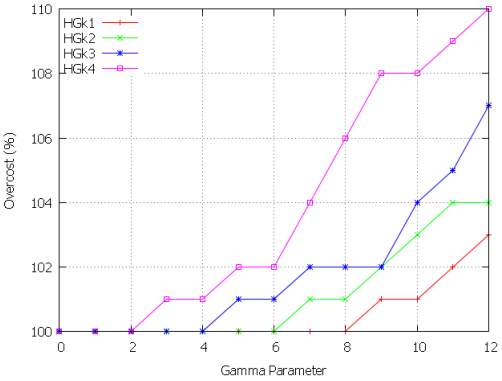

Figure 2.11: Overcosts due to the HG rule in function of parameter γ_{HG} for Net_2

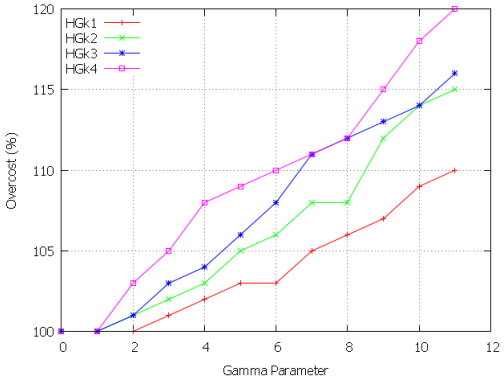

Figure 2.12: Overcosts due to the HG rule in function of parameter γ_{HG} for Net_14

Figures 2.11 and 2.12 show results for these tests. In both cases, the expected cost increase is observed, though it is more important for Net_{14}, the lowest client density instance. Indeed, for Net_2, overcosts go from 3% to 10% in the worst configuration (that is $\gamma_{HG} = 12$) whereas, for Net_{14}, they go from 10% to 20%. High density areas like Net_2 have a grid-like structure (unlike lower density areas like Net_{14}), which offers a lot of routing options. This is a possible reason for the relatively low cost increase one can observe even for the most constrained version of the HG rule on that instance.

2.4 Development of cost models for estimating future OA&M gains

This section is dedicated to an estimation of the future potential gains of taking into account OA&M constraints in network design. As mentioned in previous Sections, gains can be expected in various OA&M activities. In our approach we propose a specific macro-model for each rule, enabling us to estimate a specific source of gain. Let us emphasize that our aim is to get an insight in the gain and not to quantify it in an absolute way, which would prove irrelevant in many cases.

2.4.1 A gain from the optical splitter delocation rule: the search for failures

The specific OA&M indicator that is considered here is the time needed for troubleshooting. Troubleshooting activities include all activities consisting in monitoring the network and defaults sources. These activities are time consuming for network supervisors. In a general context, the time spent on these maintenance activities will clearly depend on the "easiness" of identifying faults and their origin. In our PON context, we claim that the time needed for finding the source of a subscriber service breakdown highly depends on the fact that its supplying equipments (i.e. splitters) are located at the subscriber's node.

Definition 2.4.1. *Let us denote by p_i^k the number of subscribers of node i that are not served by level k-or-upper splitters located at node i, we propose the following macro-model TS to estimate troubleshooting gains, defined for a solution of the PON^K and $PON^K_{OA\&M}$ problems by:*

$$TS = \sum_{k=1}^{K} \sum_{i \in V} p_i^k$$

with

$$p_i^k = \begin{cases} 0 & \forall i \in V \text{ and } k = K+1 \\ \max\left(p_i^{k+1}, d_i - (m^k \mathbf{z}_i^k - \mathbf{u}_i^{k+1}) \prod_{j=k+1}^{K} m^j\right) & \forall k = 1...K, \forall i \in V \end{cases}$$

2.4.2 A gain from the household grouping rule: the preventive maintenance rounds

The specific OA&M indicator that is considered here is the time needed for preventive maintenance. Preventive maintenance activities include all activities consisting in visiting, checking weaknesses and update/enhance the status of equipments in operations so

as to anticipate and prevent future failures. These activities are intervention workforce consuming and consequently induce significant operational costs (OPEX costs). In our PON context, the time spent on these maintenance activities will clearly depend on two main factors:

- the mean duration of an intervention on a splitter, denoted by l_{inter}
- the mean duration of the trip from a technician depot to a splitter site, denoted by l_{travel}

Definition 2.4.2. *We propose the following macro-model PM to estimate preventive maintenance workforce needs, defined for a solution of the PON^K and $PON^K_{OA\&M}$ problems by*

$$PM = \sum_{k=1}^{K} \sum_{i \in V} l_{inter} \mathbf{z}_i^k + 2 l_{travel} \mathbf{v}_i^k$$

Noting that mean duration d_{travel} can vary according to the geographical areas, we propose the following parametric model, based on one single parameter $\eta = \dfrac{2l_{travel}}{l_{inter}}$:

$$PM^{\eta} = \sum_{k=1}^{K} \sum_{i \in V} \mathbf{z}_i^k + \eta \mathbf{v}_i^k$$

2.4.3 Experimental study of OA&M rules gains

In order to assess the positive impact of OA&M rules on future OPEX, let us use the test setting as defined in section 2.3.3, for the same two instances Net_2 and Net_{14}. Thus, according to variations of parameter γ which control the strength of OA&M rules, the variations of future estimated OPEX will be shown.

The troubleshooting gains of the "Splitter Delocation" rule

Let us focus on the Splitter Delocation rule and the estimated troubleshooting gain, described in definition 2.4.1, one can expect to have while applying this rule. Problem PON^3_{SD} is solved over Net_2 and Net_{14}, using the set of delocation thresholds e^4_k defined in section 2.3.3. For each solution, TS is computed. The standard problem PON^3 is also solved for these two instances, and TS is also computed in order to have a comparison basis. Therefore, the gain or loss relative to the application of the SD rule is easily computed: $\dfrac{TS(PON^3) - TS(PON^3_{SD})}{TS(PON^3)}$.

Figures 2.13 and 2.14 give the result of this experiment. Let us recall that the lower γ_{SD} is, the stronger the SD rule is also. Therefore, it is no surprise that for very low values of γ_{SD}, which implies that most optical splitters are already put in place before

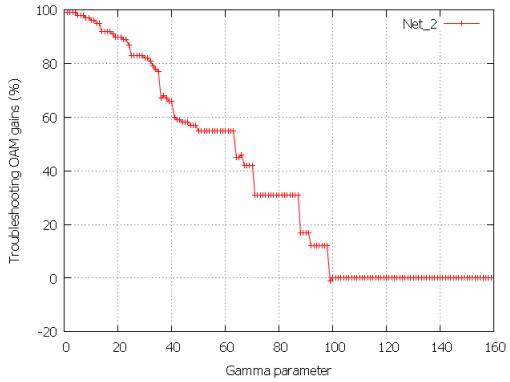

Figure 2.13: Troubleshooting gains due to the SD rule in function of parameter γ_{SD} for Net_2

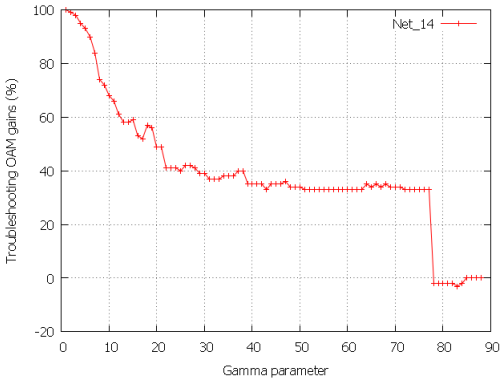

Figure 2.14: Troubleshooting gains due to the SD rule in function of parameter γ_{SD} for Net_14

optimizing thus making TS very low, the observed troubleshooting gain is high, close to 100%. Yet, another interesting fashion here is that TS does not strictly decrease as γ_{SD} increases. Indeed, there are limited fluctuations along the curve, even if, globally, the troubleshooting gain seem to decrease and tend to 0%. For Net_2, the decrease is somewhat linear, and it remains constant (close to 0%) for γ_{SD} values greater than 100. This is also true for Net_{14} for $\gamma_{SD} \geq 78$, with a non linear decrease of the gain and big gap from 0% to 30% gain around $\gamma_{SD} = 78$. This shows that the SD rule is quite ineffective when applied weakly.

However, for a strongly applied SD rule, the gain may be substantial. Indeed, for a CAPEX overcost limited to 10%, one can see (according to figures 2.9 and 2.10), that the expected troubleshooting gain given by TS can reach 55% for Net_2, and even 70% for Net_{14}. This emphasizes the fact that the SD rule enables one to loose a little now in order to save a lot later, which is the initial purpose of the OA&M considerations taken into account in the optimization.

The preventive maintenance gains of the "Household Grouping" rule

Considering the Household Grouping rule gains, linked to the future maintenance rounds cost, let us solve the problem PON^3_{HG} over Net_2 and Net_{14} with the set of thresholds HG^k_3. Computing PM^η depends on the parameter η, which could be computed using the instance properties (using the average travel time between all pair of splitter sites and considering the average intervention time). In the end, η is here to give more or less importance to the number of splitter sites compared to the total number of splitters. In order to have different point of views, let us take 3 possible values for η: 0.1, 1 and 10.

The preventive maintenance gain or loss is computed from the one observed in solutions of PON^3 and resumed by the formula: $\frac{PM^\eta(PON^3) - PM^\eta(PON^3_{SD})}{PM^\eta(PON^3)}$.

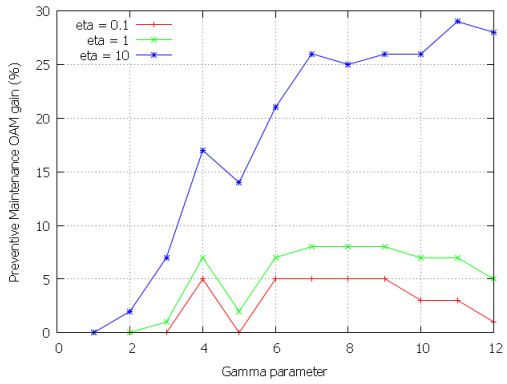

Figure 2.15: Preventive Maintenance gains due to the HG rule in function of parameter γ_{HG} for Net_2

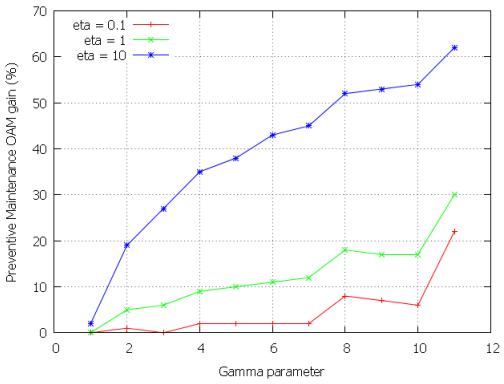

Figure 2.16: Preventive Maintenance gains due to the HG rule in function of parameter γ_{HG} for Net_14

Figures 2.15 and 2.16 show the result of the experiment. Let us recall that, contrary to the SD rule, augmenting γ_{HG} means making the HG rule stronger. Therefore, it is not surprising that the OA&M gain globally rises with γ_{HG}, even if that is not a strict increase (as shown in figure 2.15). In both cases, the gain is much higher for higher values of η, which represent areas where the travel time l_{travel} is higher than the intervention time l_{inter}. Thus, the impact of limiting the number of splitter sites that may be opened makes sense, since the biggest gain observed is around 27% for Net_2, and around 62% for Net_{14}.

Conclusion

To conclude, this chapter was dedicated to tackle the real-life issue of having both CAPEX and OPEX cost to consider in an optimization problem. In the field of telecommunications, the area that deals with the future management of a network is the OA&M. From that perspective, knowing the fact that CAPEX and OPEX are not to be optimized at the same time, because they belong to different operational entities, with different objectives, the choice was made to include these OA&M considerations in the optimization under the form of engineering rules, or constraints. These rules had to be easy to understand, with few parameters to deal with, and their effect on OA&M had to be assessed, at least in a qualitative way.

Therefore, the Splitter Delocation rule and the Household Grouping rule were created, modelled and included in the optimization. Feasibility bounds were provided to that

extent. In order to assess the rules' effects, economical macro-models were created, based on troubleshooting activities and future preventive maintenance rounds. These models are easy to compute and simple to understand, with very few parametrization to do. However, their purpose is only qualitative, and their role is limited to assessment.

From this approach, many other engineering rules may have been chosen, as well as many other economical macro-models. This work could be extended by adding these new rules and their economical counterpart. The main point of this study is, in definitive, to propose a methodology for dealing with future OPEX costs in a real-life optimization context, and we believe that it could be applied to other network optimization problems.

This work has been published in the Journal of Applied Operational Research, Volume 4(3) in September 2012 by Matthieu Chardy and Cédric Hervet under the title *Passive optical network design under operations administrations and maintenance considerations*, pages 152–172.

Instance	Architecture 1 (64)				Architecture 2 (2:32)				Architecture 3 (2:8:4)			
	$C\mathbf{z}$	$c\mathbf{f}$	gap	Time	$C\mathbf{z}$	$c\mathbf{f}$	gap	Time	$C\mathbf{z}$	$c\mathbf{f}$	gap	Time
Net_1	572424	84632	0.92%	550	638043	77931.5	1.22%	550	1302789	65449.7	0.74%	550
Net_2	20332	2574.9	0.00%	0.8	21528	2972.2	0.00%	4.8	36524	9467.6	0.03%	550
Net_3	150926	20076.6	0.37%	550	162081	20578.8	0.18%	550	330694	22706.6	0.10%	550
Net_4	311236	36737.5	0.62%	550	339618	36464.1	0.53%	550	697682	34020.4	0.44%	550
Net_5	195500	28758.1	0.75%	550	213210	28082.5	0.46%	550	431250	30149.6	0.24%	550
Net_6	273700	37076.5	0.83%	550	300702	34443.3	0.43%	550	611340	35160.4	0.34%	550
Net_7	142324	19224.4	0.41%	550	154077	18935.5	0.33%	550	311880	23226.3	0.14%	550
Net_8	304198	126051.3	2.65%	550	337548	113530.8	5.35%	550	669760	93041.1	1.95%	550
Net_9	113390	47051.5	2.63%	550	123993	36630.1	1.97%	550	247089	30222.9	1.21%	550
Net_{10}	229126	95026.3	2.46%	550	257784	82444.4	4.47%	550	510370	70259.4	1.65%	550
Net_{11}	30498	11877	2.75%	550	32223	8585.7	1.44%	550	62813	7718.9	0.79%	550
Net_{12}	173604	77463.7	2.14%	550	196995	65795.7	3.88%	550	394519	54338.1	1.28%	550
Net_{13}	70380	19394.3	1.93%	550	76659	15792.6	1.41%	550	155480	12708.9	0.65%	550
Net_{14}	14076	3297.3	0.00%	9.4	14904	2873.6	0.00%	107	28566	3652.9	0.15%	550

Table 2.3: Results of the PON^K problem for 3 architectures over the set of instances

Chapter 3

Optimization of Passive Optical Network deployments with cabling constraints in an arborescence

Introduction

In an operational context, there are many engineering aspects to be considered. Some of them are not integrated in optimization models because they can be negligible or too complex to model, or both. For the passive optical network deployment problem, cabling constraints are among those that may have a non negligible impact on both solution cost and feasibility and that can be hard to integrate in existing models. Indeed, in the literature, there are a few references that tackled this specific issue. In his Ph.D. thesis [50], Trampont tried to integrate this aspect in his models for generic non oriented graphs. However, it proved to be hardly tractable if directly put into Integer Linear Programs. Therefore he proposed complex heuristics to modify a relaxed solution with no cables in order to find a good "cabled one". The work from Kim et al. [34] is also worth being mentioned: such constraints are integrated, but in a specific context where the graph over which the optimization is made is an arborescence. In their work, they also point out the fact that cabling constraints make the problem hard to solve and they propose relaxations in order to find upper and lower bounds for the problem, as well as a dedicated heuristic.

Moreover, from an industrial point of view, the optimization tool developed at Orange Labs for the field PON network deployments, must take into account almost every engineering rules, including cabling constraints. Of course, the real-life size of the problem (from 1000 to 5000 nodes) and the numerous and complex engineering rules make the whole problem almost impossible to solve all at once. Therefore, a decomposition of the problem had to be considered in order to solve several tractable sub-problems sequentially,

thus enabling the tool to propose good and valid solutions while loosing the guarantee of optimality. Today, cabling constraints are integrated only once the routing-of-fibers step is done. And as mentioned above, this process is highly heuristic and it can lead to cost sub-optimal deployments.

In this chapter, we aim to tackle the passive optical access network deployment problem with cabling constraints in an arborescence. Based on the literature, it seemed to us that this specific version of the problem was a propitious field for many theoretical and practical improvements. Indeed, to the best of our knowledge, there is no solution available for solving that problem to optimality for real size instances. Therefore, the first objective of this chapter is to propose a new model for the PON problem with cabling constraints. Then, from the literature, the second objective is to extend and prove some properties of this problem. Finally, our last objective will be to propose a dedicated labeling algorithm to solve the problem to optimality on real-life instances so that it can be integrated in an operational decision-making tool. This algorithm will be compared to exact branch and bound solving approaches.

3.1 Modeling of the problem with cabling constraints in an arborescence

3.1.1 An arc based formulation for the Passive Optical Network design with cables constraints problem in an arborescence

In this section, we aim to propose a new model for the optical deployment in an arborescence, different from the one proposed by Kim et al. [34] (which is presented in section 1.1.4) and based on Trampont's model.

Keeping the notations introduced in Section 1.1.4, let us define another set of variables for modeling the problem. We denote by $\Gamma^{-1}(i)$ the unique father node of node $i \in V^*$ and by $\Gamma^+(i)$ the set of sons of node i (which is an empty set if i is a leaf). As for decision variables, we denote by $\mathbf{z}_i^k$ the number of splitters of level k (for $k \in \{1,2\}$) installed at node $i \in V$ and by $\mathbf{f}_{ij}^k$ the number of fibers of level k (for $k \in \{1,2,3\}$) routed along the arc $(i,j) \in A$. Variables $z_{olt}^k, k \in \{1,2\}$ are introduced in this chapter for convenience but are equal to 0 for any deployment. We keep the variables $\mathbf{c}_{ij}^l, (i,j) \in A$ as defined in the previous section. We thus propose a reformulation of the PON deployment of a 2 level architecture optimization problem:

$$
(P^2_{tree})\begin{cases}
\min\limits_{\mathbf{z},\mathbf{f},\mathbf{c}} & \sum\limits_{l\in L}\sum\limits_{(i,j)\in A} g^l_{ij}\mathbf{c}^l_{ij} + \sum\limits_{k=1}^{2}\sum\limits_{i\in V^*} C^k\mathbf{z}^k_i \\
\text{s.t.} & \mathbf{f}^1_{\Gamma^{-1}(i)i} = \mathbf{z}^1_i + \sum\limits_{j\in\Gamma^+(i)} \mathbf{f}^1_{ij},\ \forall i\in V^*, & (3.1)\\
& \mathbf{f}^2_{\Gamma^{-1}(i)i} + m^1\mathbf{z}^1_i \geq \mathbf{z}^2_i + \sum\limits_{j\in\Gamma^+(i)} \mathbf{f}^2_{ij},\ \forall i\in V^*, & (3.2)\\
& \mathbf{f}^3_{\Gamma^{-1}(i)i} + m^2\mathbf{z}^2_i \geq d_i + \sum\limits_{j\in\Gamma^+(i)} \mathbf{f}^3_{ij},\ \forall i\in V^*, & (3.3)\\
& \sum\limits_{k=1}^{3}\mathbf{f}^k_{ij} \leq \sum\limits_{l\in L} b^l\mathbf{c}^l_{ij},\ \forall (i,j)\in A, & (3.4)\\
& \sum\limits_{l\in L}\mathbf{c}^l_{ij} \leq 1,\ \forall (i,j)\in A, & (3.5)\\
& \mathbf{f}^k_{ij}\in\mathbb{N},\ \forall (i,j)\in A,\ k\in 1..3,\\
& \mathbf{z}^k_i\in\mathbb{N},\ \forall i\in V^*,\ k\in 1..2,\\
& \mathbf{c}^l_{ij}\in\{0,1\},\ \forall (i,j)\in A,\ \forall l\in L.
\end{cases}
$$

Remark 3.1.1. *From this model, one can straightforward derive a similar formulation for a single level architecture problem. Variables* $\mathbf{f}^3_{ij}$ *and* $\mathbf{z}^2_i$ *can be removed and the problem* (P^1_{tree}) *can be formulated as follows:*

$$
(P^1_{tree})\begin{cases}
\min\limits_{\mathbf{z},\mathbf{f},\mathbf{c}} & \sum\limits_{l\in L}\sum\limits_{(i,j)\in A} g^l_{ij}\mathbf{c}^l_{ij} + \sum\limits_{i\in V^*} C^1\mathbf{z}^1_i \\
s.t. & \mathbf{f}^1_{\Gamma^{-1}(i)i} = \mathbf{z}_i + \sum\limits_{j\in\Gamma^+(i)} \mathbf{f}^1_{ij},\ \forall i\in V^*, & (3.6)\\
& \mathbf{f}^2_{\Gamma^{-1}(i)i} + m\mathbf{z}_i \geq d_i + \sum\limits_{j\in\Gamma^+(i)} \mathbf{f}^2_{ij},\ \forall i\in V^*, & (3.7)\\
& \sum\limits_{k=1}^{2}\mathbf{f}^k_{ij} \leq \sum\limits_{l\in L} b^l\mathbf{c}^l_{ij},\ \forall (i,j)\in A, & (3.8)\\
& \sum\limits_{l\in L}\mathbf{c}^l_{ij} \leq 1,\ \forall (i,j)\in A, & (3.9)\\
& \mathbf{f}^k_{ij}\in\mathbb{N},\ \forall (i,j)\in A,\ k\in 1..2,\\
& \mathbf{z}_i\in\mathbb{N},\ \forall i\in V^*,\\
& \mathbf{c}^l_{ij}\in\{0,1\},\ \forall (i,j)\in A,\ \forall l\in L.
\end{cases}
$$

Remark 3.1.2. *Investigating the formulation* (PON^2_{Kim}) *of Kim et al. (see Section 1.1.4), we assume that its relative difficulty to tackle larger instances came from its symmetries. Indeed, they are using a path based formulation to explicitly describe all the*

physical optical branching possibilities from the OLT to any splitters, from any splitter to any other splitter, and so on until the demand node is reached. On the contrary, P^1_{tree} *and* P^2_{tree} *relie on an arc based formulation, thus getting rid of the symmetry related to physical optical branching options leading to the same fiber cost (see Figure 2.2).*

3.1.2 Experimental comparison of formulations

Let us now compare these formulations on an experimental basis. We generated several instances in which we solved the Passive Optical Network design problem with cabling constraints in an arborescence, with a single level architecture, and a two level one. We generated instances in the size range $\{10, 20, 30, ..., 100\}$. For each instance size, we randomly generated 10 instances. For a given size, we report for the average solving time ("CPU Time (sec)" columns) over the 10 instances of this size, as well as the average gap ("gap (%)" columns). Instances are solved by the mean of the solver CPLEX 12.2 with a time limit set to 1000 seconds for each run. Note that when no gap is reported, it means that all instances were solved to optimality within the time limit for the considered size.

Instance size	P^2_{tree} Formulation		Kim et al. formulation	
	CPU time (sec)	gap (%)	CPU time (sec)	gap (%)
10	0.02	-	0.02	-
20	0.09	-	0.30	-
30	0.41	-	133	-
40	2.89	-	754	1.21
50	5.23	-	1000	1.75
60	138	0.02	1000	2.07

Table 3.1: Comparison of the two MILP formulations for a 2 level PON deployment

Results of the experiment are reported in Tables 3.1 and 3.2. Two major observations must be made:

- (P^1_{tree}) and (P^2_{tree}) formulations outperform those from Kim et a.l when a branch and bound solving approach is considered. This was to be expected since the formulation of Kim et al. introduces a lot of variables and symmetry.
- both approaches are very limited in terms of size of instances solved to optimality.

As a conclusion, both formulations prove ineffective for solving real-life instances. Thus, in the next section, we will try to design a new algorithm that can tackle larger instances of the problem.

Instance size	P^1_{tree} Formulation		Kim et al. formulation	
	CPU time (sec)	gap (%)	CPU time (sec)	gap (%)
10	0.008	-	0.016	-
20	0.035	-	0.036	-
30	0.181	-	0.248	-
40	0.244	-	1.523	-
50	0.349	-	9.121	-
60	0.536	-	67.72	0.03
70	1.195	-	168.1	0.06
80	3.631	-	438.1	0.19
90	13.44	-	718.3	0.39
100	49.78	0.01	885.3	0.66

Table 3.2: Comparison of the two MILP formulations for a single level PON deployment

3.2 Study of the problem's properties

The paper of Kim et al. contains properties of the problem's optimal solutions. Some of them could be extended. Since it is always observed on the field, we assume in the following that cable costs increase with respect to their capacities.

All the definitions and propositions of this section are given (only) for (P^2_{tree}) as they embed the similar results for (P^1_{tree}).

Definition 3.2.1. *The sub-tree of $G(V, A)$ induced by node $j \in V$ is denoted by $T(j)$, j being thus the root node of $T(j)$.*

Definition 3.2.2. *Let us introduce the notion of configuration of a node $j \in V$, denoted by $s_j = (\hat{\mathbf{z}}, \hat{\mathbf{f}}, \hat{\mathbf{c}})$, where:*

- $\hat{\mathbf{z}}$ *denotes the splitter variables vector restricted to the nodes of the induced sub-tree $T(j)$,*
- $\hat{\mathbf{f}}$ *denotes the fiber variables vector restricted to the arcs of the induced sub-tree $T(j)$,*
- $\hat{\mathbf{c}}$ *denotes the cable variables vector restricted to the arcs of the induced sub-tree $T(j)$.*

s_j is a feasible configuration iif it verifies the set of constraints of (P^2_{tree}).

Remark 3.2.1. *A feasible configuration of the OLT node is a feasible solution of (P^2_{tree}). However, a feasible configuration of node $j \in V^*$ does not necessary correspond to a solution of (P^2_{tree}) restricted to the induced sub-tree $T(j)$.*

3.2.1 Bounds on the number of splitters to be placed at every node

Proposition 3.2.1. *Focusing on the cumulative number of splitters (of level 1 or 2) installed in each sub-tree, there always exists an optimal solution which satisfies:*

$$\sum_{i \in T(j)} \mathbf{z}_i^2 \geq \left\lfloor \frac{\sum_{i \in T(j)} d_i}{m^2} \right\rfloor \quad \forall j \in V^* \tag{3.10}$$

$$\sum_{i \in T(j)} \mathbf{z}_i^1 \geq \left\lfloor \frac{\sum_{i \in T(j)} \mathbf{z}_i^2}{m^1} \right\rfloor \quad \forall j \in V^* \tag{3.11}$$

Proof. Let x be an optimal solution from (P^2_{tree}) such that $\sum_{i \in T(j)} \mathbf{z}_i^2 = \left\lfloor \frac{\sum_{i \in T(j)} d_i}{m^2} \right\rfloor - \tau$ with $\tau \geq 1$. This implies that there are at least $m^2\tau$ level 2 fibers coming from the unique father node of j, along the arc $(\tau^-(j)j)$. Let us follow these $m^2\tau$ level 2 fibers on the same reverse path (it is possible since G is a tree) until we reach the τ first encountered level 1 splitters along the path. Let $j_1, ..., j_K$ the corresponding installation nodes (all different except if $K = \tau$) . We build x' from x by "bottomizing" these splitters to node j, that is to say:
$\forall n = 1, ..., K$:

- let $z_n \in N = \min(z^2_{j_n}; \tau - \sum_{k=1,...,n-1} z^2_{j_k})$,
- we change the place of the level 2 splitters: $z'^2_{j_n} = z^2_{j_n} - z_n$ and $z'^2_j = z^2_j + z_n$,
- we adapt the level 2 fiber flows $\forall a \in path(j_n, j) : \ f'^2_a = f^2_a - m^2 z_n$ as well as the level 1 fiber ones $\forall a \in path(j_n, j) : \ {f'^1}_a = f^1_a + z_n$.

Then we compute suitable cable size in arcs whose fiber flows were changed: $\forall a \in path(j_K, j),\ l_a = \arg\min_{l \in L} b^l \ s.t. \sum_{k \in 1,...,3} f'^k_a \leq b^l$, setting $c^{l_a}_a = 1$ and $c^l_a = 0,\ \forall l \neq l_a \in L$. Noting that $\sum_{k \in 1,...,3} f'^k_a \leq \sum_{k \in 1,...,3} f^k_a$, we ensure that x' is feasible and potentially of lower cost. Therefore, (3.10) is true for some optimal solutions.
(3.11) can, in addition, be proven by the same reasoning, only considering that fibers originated from level 1 splitters must serve level 2 splitters instead of demands. □

Corollary 3.2.1. *Focusing on the number of splitters (of level 1 or 2) installed in each*

node, there always exists an optimal solution which satisfies:

$$\mathbf{z}_j^2 \geq \left\lfloor \frac{\sum_{i \in T(j)} d_i}{m^2} \right\rfloor - \sum_{i \in T(j') | j' \in \Gamma^+(j)} \mathbf{z}_i^2, \ \forall j \in V^* \tag{3.12}$$

$$\mathbf{z}_j^1 \geq \left\lfloor \frac{\sum_{i \in T(j)} \mathbf{z}_i^2}{m^1} \right\rfloor - \sum_{i \in T(j') | j' \in \Gamma^+(j)} \mathbf{z}_i^1, \ \forall j \in V^* \tag{3.13}$$

Corollary 3.2.1 is worth of interest, as it gives a lower bound of the optimal number of splitters to install on a node as a function of the number of splitters of the same level installed in the sub-tree originating from this node. Let us now focus on an finding an upper bound for the number of splitters at a node.

Proposition 3.2.2. *Focusing the cumulative number of splitters (of level 1 or 2) installed in each sub-tree, any optimal solution satisfies:*

$$\sum_{i \in T(j)} \mathbf{z}_i^2 \leq \sum_{i \in T(j)} \left\lceil \frac{d_i}{m^2} \right\rceil, \ \forall j \in V \tag{3.14}$$

$$\sum_{i \in T(j)} \mathbf{z}_i^1 \leq \sum_{i \in T(j)} \left\lceil \frac{\mathbf{z}_i^2}{m^1} \right\rceil, \ \forall j \in V \tag{3.15}$$

Proof. Let us consider the deployment policy consisting in serving all demand nodes by co-located level 2 splitters, thus installing $\left\lceil \frac{d_i}{m^2} \right\rceil$ level 2 splitters on every demand node i. This is obviously the most consuming policy in terms of level 2 splitters (no sharing of any level 2 splitter among demand nodes), thus defining an upper bound for this type of splitters to be installed in any sub-tree, hence (3.14). Note that the fact can prove unfeasible with respect to cabling constraints (3.4) and (3.5), but this fact can only reduce these bonds. The same reasoning applies to level 1 splitters for deriving (3.15). □

Corollary 3.2.2. *In any optimal solution, an upper bound for the number of optical splitters (of level 1 or 2) installed at each node is given by:*

$$\mathbf{z}_j^2 \leq \sum_{i \in T(j)} \left\lceil \frac{d_i}{m^2} \right\rceil - \sum_{i \in T(j') | j' \in \Gamma^+(j)} \mathbf{z}_i^2, \ \forall j \in V^* \tag{3.16}$$

$$\mathbf{z}_j^1 \leq \sum_{i \in T(j)} \left\lceil \frac{\mathbf{z}_i^2}{m^1} \right\rceil - \sum_{i \in T(j') | j' \in \Gamma^+(j)} \mathbf{z}_i^1, \ \forall j \in V^* \tag{3.17}$$

Corollary 3.2.2 is worth of interest, as it can give an upper bound of the optimal number of splitters to install on a node as a function of the number of splitters of the same level installed in the sub-tree originated from its node. The previous corollaries can be combined in the following proposition.

Proposition 3.2.3. *On any leaf node j, the optimal number of optical splitter to be installed is such that:*

$$\left\lfloor \frac{d_j}{m^2} \right\rfloor \leq \mathbf{z}_j^2 \leq \left\lceil \frac{d_j}{m^2} \right\rceil \tag{3.18}$$

$$\left\lfloor \frac{\mathbf{z}_j^2}{m^1} \right\rfloor \leq \mathbf{z}_j^1 \leq \left\lceil \frac{\mathbf{z}_j^2}{m^1} \right\rceil \tag{3.19}$$

Proof. This is a straightforward consequence of corollaries 3.2.1 and 3.2.2. □

Corollary 3.2.3. *There are at most 2 interesting possible splitter numbers per splitter level for a leaf leaf node (there is only one if d_j, respectively $\mathbf{z}_j^2$, are a multiple of m^2, respectively m^1).*

3.2.2 A cutting rule for ensuring cabling feasibility

Definition 3.2.3. *We associate to any feasible configuration $s_j = (\hat{\mathbf{z}}, \hat{\mathbf{f}}, \hat{\mathbf{c}})$ (of any node $j \in V$) a vector of remaining demand in fibers. The vector of remaining demand is denoted $\hat{d}^k(s_j) \in \mathbb{N}$ and defined by the following recursion:*

- *initialization (if j is a leaf):*

$$\hat{d}^3(s_j) = \max\left\{d_j - m^2\hat{\mathbf{z}}_j^2(s_j),\, 0\right\} \tag{3.20}$$

$$\hat{d}^2(s_j) = \max\left\{\hat{\mathbf{z}}_j^2(s_j) - m^1\hat{\mathbf{z}}_j^1(s_j),\, 0\right\} \tag{3.21}$$

$$\hat{d}^1(s_j) = \hat{\mathbf{z}}_j^1(s_j) \tag{3.22}$$

- *recursive function (if $i \in V$, not being a leaf):*

$$\hat{d}^3(s_j) = \max\left\{\sum_{i\in\Gamma^+(j)} \hat{d}^3(s_i) - m^2\hat{\mathbf{z}}_j^2(s_j) + d_j;\, 0\right\}, \tag{3.23}$$

$$\hat{d}^2(s_j) = \max\left\{\sum_{i\in\Gamma^+(j)} \hat{d}^2(s_i) - m^1\hat{\mathbf{z}}_j^1(s_j) + \hat{\mathbf{z}}_j^2(s_j);\, 0\right\}, \tag{3.24}$$

$$\hat{d}^1(s_j) = \sum_{i\in\Gamma^+(j)} \hat{d}^1(s_i) + \hat{\mathbf{z}}_j^1(s_j) \tag{3.25}$$

Proposition 3.2.4. *A feasible configuration $s_j = (\hat{\mathbf{z}}, \hat{\mathbf{f}}, \hat{\mathbf{c}})$ of node $j \in V^*$ such that:*

$$\sum_{k=1}^{3} \hat{d}^k(s_j) > \max_{l\in L} b^l \tag{3.26}$$

does not correspond to a solution of (P^2_{tree}) restricted to the induced sub-tree $T(j)$ (see. Remark 3.2.1).

Proof. This is straightforward when noticing that the whole cumulative remaining demand of node j, $\sum_{k=1}^{3} \hat{d}^k(s_j)$, should be routed to j from its unique father node $\Gamma^-(j)$, which would make constraints (3.4) and (3.5) of problem (P^2_{tree}) violated for at least the arc $(\Gamma^-(j)j)$. □

3.2.3 Design of a dominance rule on the sets of feasible configuration

Before entering the details of the dominance rule we aim to introduce, let us introduce some definitions first.

Definition 3.2.4. *We associate a cost to any feasible configuration $s_j = (\hat{\mathbf{z}}, \hat{\mathbf{f}}, \hat{\mathbf{c}})$ (of any node $j \in V$) defined by the following recursion:*

- *initialization (if j is a leaf):*

$$cost(s_i) = \sum_{k=1}^{2} C^k \hat{\mathbf{z}}^k_j(s_j) \tag{3.27}$$

- *recursive function (if $i \in V$, not being a leaf):*

$$cost(s_j) = \sum_{k=1}^{2} C^k \hat{\mathbf{z}}^k_j(s_j) + \sum_{i \in \Gamma^+(j)} \left(cost(s_i) + \sum_{l \in L} g^l_{ji} \hat{\mathbf{c}}^l_{ji}(s_j) \right) \tag{3.28}$$

Let us now introduce a dominance relation on the set of feasible configurations of any node $j \in V$.

Definition 3.2.5. *Let s_j and s'_j two feasible configurations of node $j \in V$. Then s_j dominates s'_j, noted by $s_j \succeq s'_j$, if*

$$\begin{cases} \hat{d}^k(s_j) \leq \hat{d}^k(s'_j), \ \forall k \in \{1,2,3\} \\ cost(s_j) \leq cost(s'_j) \end{cases}$$

and that at least one of these inequations is strictly verified.

Proposition 3.2.5. *Removing all dominated configurations from the set of feasible configuration of $j \in V$ does not remove all optimal solution vectors of (P^2_{tree}) restricted to $T(j)$.*

Proof. Let s_j and s'_j be two feasible configurations of node j. Let s_j correspond to an optimal solution vector x^* of (P^2_{tree}) restricted to $T(j)$, and $s'_j \succeq s_j$. Our objective is to build an optimal solution x' of (P^2_{tree}) from s'_j. Let extend s'_j with the x^* solution vector

restricted to $G(V,A) \setminus T(j)$. As $\hat{d}^k(s_j) \leq \hat{d}^k(s'_j)\ \forall k \in \{1,2,3\}$, then $(f^{*^k}_{\Gamma^-(j),j})_{k=1,\dots,3}$ are compliant with s'_j remaining demand. Then x' is a feasible solution for (P^2_{tree}). As $cost(s'_j) \leq cost(s_j)$ (the only part of the total cost that can differ between x' and x^*), we have x' optimal. □

3.3 Design of a labelling algorithm for solving the problem to optimality

In Section 3.2, we developed the theoretical material that is used in the design of our exact labelling algorithm for solving (P^2_{tree}) to optimality.

Let us describe the general scheme of our labelling algorithm, which consists in recursively build the set of feasible configurations of each node, starting from the leaves until reaching the OLT. Results of the previous section are used to reduce the number of configuration that we explore and propagate during the algorithm.

The first step of the algorithm is be to sort V according to the nodes depth in the tree T, in a decreasing order. Let this new set be $\tilde{V}$. The second step is to sequentially explore the nodes of $\tilde{V}$, and 3 cases can occur:

- The node is a leaf: at most 2 feasible configurations are generated thanks to corollaries 3.2.1 and 3.2.2, and if any of them is infeasible due to cable limitation, it is removed (the cut of Proposition 3.2.4).

- The node is not a leaf: we build all the feasible configurations from the Cartesian product of the sets of configurations of its son nodes for details on this building). From any combination of a feasible configuration of each son node, at most 2 feasible configurations are kept thanks to corollaries 3.2.1 and 3.2.2. Endly, the dominance rule given by definition 3.2.5 is applied to remove dominated feasible configurations.

- The node is the root of the tree (OLT): build all the feasible configurations from the Cartesian product of the sets of configurations of its son nodes. Since no optical splitters can be put on the OLT, the only configurations we need to consider are those with null remaining demands and splitter variables.

The third step consists in selecting a configuration of minimal cost (see. Definition 3.2.4) at the OLT node: this configuration correspond to an optimal solution of (P^2_{tree}).

We provide the pseudo-code of the algorithm in a more formal way in Algorithm 3.

Algorithm 3 Labelling algorithm for solving (P^2_{tree}) to optimality.

Require: $G(V, A)$.

1: Create $\tilde{V}$ by sorting V according to their depth in a decreasing order.
2: **for** $j = 1$ to $|\tilde{V}|$ **do**
3: **if** j is a leaf node **then**
4: Build the two feasible configurations compliant with corollaries 3.2.1 and 3.2.2.
5: **else**
6: `Extend`(j): build the set of feasible configurations from the Cartesian product of the sets of feasible configurations of the son nodes of j, filtered by the use of corollaries 3.2.1 and 3.2.2. Function `Extend` is detailed in Algorithm 4.
7: **end if**
8: Apply Proposition 3.2.4 to remove configurations.
9: Apply the dominance rule from Definition 3.2.5 to eliminate dominated configurations.
10: **end for**
11: Consider the OLT node: configurations s_{olt} such that $\hat{d}^2(s_{olt}) \neq 0$ or $\hat{d}^3(s_{olt})$ or $\hat{z}^2_{olt} \neq 0$ or $\hat{z}^1 \neq 0$ are removed.
12: **return** $\min_{s_{olt}} cost(s_{olt})$

Remark 3.3.1. *The fact that the algorithm is an exact solution method is ensured by Proposition 3.2.5*

Remark 3.3.2. *The practical efficiency of labelling algorithms (in terms of computation time or memory used) strongly relies on a smart management of in its coding. We wanted to stress the fact that the implementation of the algorithm was in practice done in a more efficient way as it is synthesized above for sake of understanding.*

To conclude this section, we illustrate the algorithm on an example on Figure 3.1 where, for the sake of clarity, only one splitter level is considered. Since it is almost to represent configurations, we use labels on the figure, denoted $label_s$ when associated to a configuration s, such that $label_s = (\hat{d}^1(s), \hat{d}^2(s), \hat{d}^3(s), cost(s))$. Starting by the two leaf nodes at the bottom right of the tree, labels are generated. The node with 8 demands only generates one label since the splitter capacity equals 8. The cost is shown as the sum of the installed splitter and the required cable. At their father node, the orange rectangle shows the result of the Cartesian product of all configurations. Since the node with 8 demands only have one label, 2 labels are generated by the Cartesian product. The 10 demands of the father node are then added in the labels, and the two possible configurations are generated from each label. It results in 4 possible configurations. All

are feasible regarding the cable size, but the last two configurations are dominated by the second one. They can be removed. As the node is a son node of the root, the first solution can not be kept, as for the other son node of the root. In the end, only one configuration is possible: it is the optimal solution of the problem.

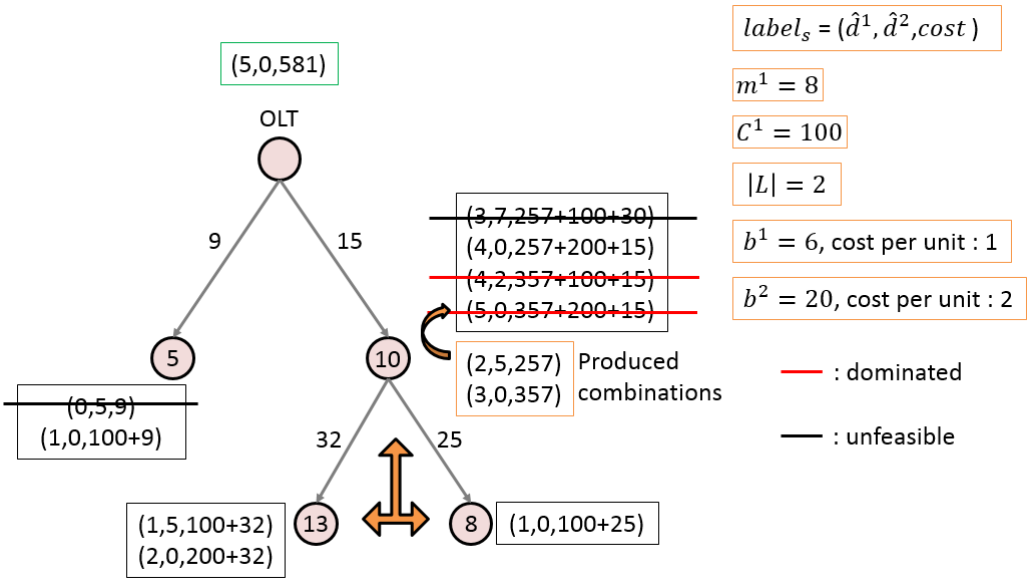

Figure 3.1: An illustration of the labelling algorithm on a little instance with only one splitter level

3.4 Experimental testing of the algorithm

The objective of this Section is to assess the efficiency of the method to solve real-life instances, which implies between 1000 and 5000 nodes on average. To that extent, we randomly generated tree graphs of various sizes, and for each size of graph (being a multiple of 10), we generated 50 instances that we solved with Algorithm 3. For each instance solved, we export the solving time and the number of labels that were kept at each node after the appliance of both the cut rule and the dominance rule. All results are displayed on Figures 3.2 and 3.3.

The results clearly indicates the tractability of the dynamic programming approach, which solves real-size instances. Indeed, even the largest instances that were considered here are solved within 50 seconds. Even though the solving time starts to increase exponentially, one could hope to solve larger instances. In fact, tests showed that the memory limitations allowed us to solve instances between 5000 and 6000 nodes, in less than half an hour. Note that the range of solving times for a given instance size also increases.

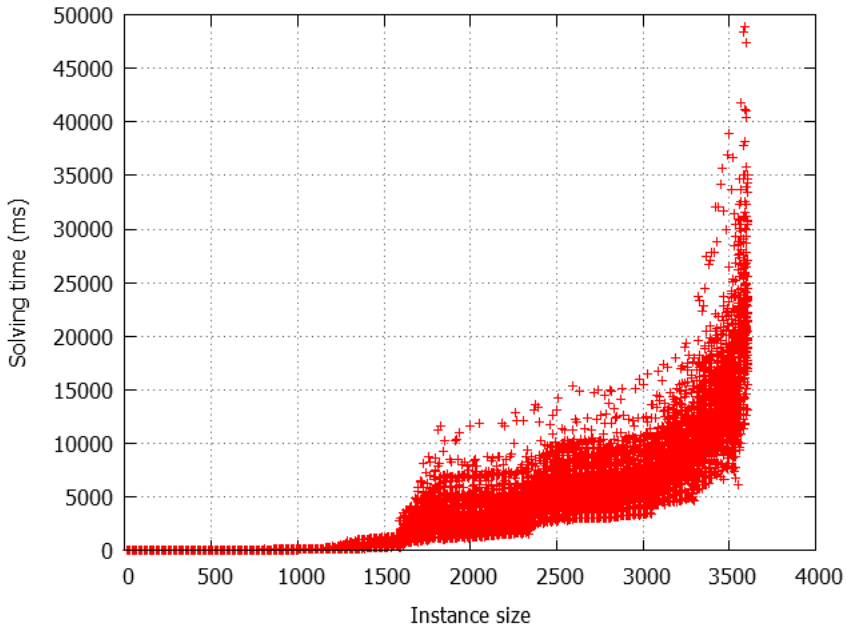

Figure 3.2: Solving times for all tested instances in function of the instance size

Indeed for the largest instance size showed here, the solving time is between 10 seconds and 50 seconds. This exponential behavior was to be expected since Algorithm 3 does not avoid the curse of dimensionality, it only delays it.

Figure 3.3 shows the sum of relevant labels at each node generated throughout the whole algorithmic process. Contrary to the solving time, it increases linearly. This emphasizes the stability and the delaying of the combinatorial explosion provided by properties of Section 3.2.

Conclusion

To conclude, this chapter explored a specific version of the Passive Optical Network design problem in tree-graphs that takes cabling constraints into account. From the initial work of Kim et al., we proposed another model for this problem, from which we were able to derive some properties. These properties were then used to recursively characterize optimal solutions, by the mean of the tree structure of the graph. A labeling algorithm was then designed to solve the problem to optimality. All solving approaches were then compared, showing that our new modeling seems to perform better than the one proposed by Kim et al., but the major result being the fact the labeling algorithm outperforms branch-and-bound approaches based on both models.

As such, these results are encouraging and one can imagine various ways of improving

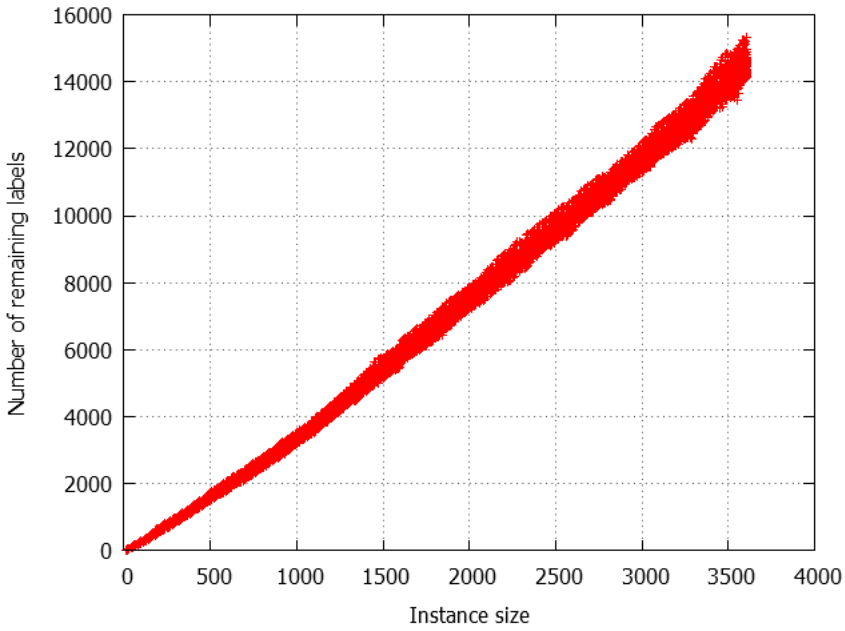

Figure 3.3: Relevant labels for all tested instances in function of the instance size

the model so it could get closer to field expectations, while improving the labeling algorithm as well so it can take these model improvements into account. We chose to leave this research way opened so we could focus the major issue that field deployments teams are struggling with: the uncertainty on fiber demand.

Algorithm 4 `Extend` function

Require: Let P the number of sons of node j : $son_1, ...son_P$ the nodes sons with their set of configurations $CONFIG_{son_1}, ..., CONFIG_{son_P}$.

Require: Let $CONFIG_j = \emptyset$ the set of configurations of j to be built.

Require: Let $s_j = null$ (one configuration)

for $(s_{son_1}, ..., s_{son_P}) \in (CONFIG_{son_1} \times ... \times CONFIG_{son_P})$ **do**
  Initialize $s_j = \left(\hat{z}, \hat{f}, \hat{c}\right)$.
  for $son_i \in (s_{son_1}, ..., s_{son_P})$ **do**
    $\hat{f}^1_{j,son_i} = \hat{d}^1(s_{son_i})$; $\hat{f}^2_{j,son_i} = \hat{d}^2(s_{son_i})$; $\hat{f}^3_{j,son_i} = \hat{d}^3(s_{son_i})$.
    $l_{j,son_i} = \arg\min_{l \in L} b^l \ s.t. \sum_{k \in 1,...,3} \hat{f}^k_{j,son_i} \leq b^l$.
    $\hat{c}^{l_{j,son_i}}_{j,son_i} = 1$.
    for $l \neq l_{j,son_i} \in L$ **do**
      $\hat{c}^l_{j,son_i} = 0$.
    end for
    for $(a \in A_{son_i} \mid T(son_i) = (V_{son_i}, A_{son_i}))$ **do**
      $\hat{f}^1_a = \hat{f}^1_a(s_{son_i})$; $\hat{f}^2_a = \hat{f}^2_a(s_{son_i})$; $\hat{f}^3_a = \hat{f}^3_a(s_{son_i})$.
      for $l \in L$ **do**
        $\hat{c}^l_a = \hat{c}^l_a(s_{son_i})$.
      end for
    end for
    for $(i' \in V_{son_i} \mid T(son_i) = (V_{son_i}, A_{son_i}))$ **do**
      $\hat{z}^1_{i'} = \hat{z}^1_{i'}(s_{son_i})$; $\hat{z}^2_{i'} = \hat{z}^2_{i'}(s_{son_i})$.
    end for
  end for
  for $z2 \in N$ **do**
    if $\left(\left\lfloor \frac{\sum_{i' \in T(j)} d_{i'}}{m^2} \right\rfloor - \sum_{i' \in T(son_i) | son_i \in \Gamma^+(j)} \hat{\mathbf{z}}^2_{i'} \leq z2 \leq \sum_{i' \in T(j)} \left\lceil \frac{d_{i'}}{m^2} \right\rceil - \sum_{i' \in T(son_i) | son_i \in \Gamma^+(j)} \hat{\mathbf{z}}^2_{i'}\right)$ **then**
      $\hat{z}^2_j = z2$.
      for $z1 \in N$ **do**
        if $\left(\left\lfloor \frac{\sum_{i' \in T(j)} \hat{\mathbf{z}}^2_{i'}}{m^1} \right\rfloor - \sum_{i' \in T(son_i) | son_i \in \Gamma^+(j)} \hat{\mathbf{z}}^1_{i'} \leq z1 \leq \sum_{i' \in T(j)} \left\lceil \frac{\hat{\mathbf{z}}^2_{i'}}{m^1} \right\rceil - \sum_{i' \in T(son_i) | son_i \in \Gamma^+(j)} \hat{\mathbf{z}}^1_{i'}\right)$ **then**
          $\hat{z}^1_j = z1$
          $CONFIG_j = CONFIG_j \cup \{s_j\}$
        end if
      end for
    end if
  end for
end for
return $CONFIG_j$

Chapter 4

The modeling of the passive optical network deployment optimization problem under demand uncertainty through robust optimization

Introduction

Chapters 2 and 3 focused on engineering-oriented aspects of the passive optical network deployment. However, there is an issue that is recurrent in network design problems: the uncertainty on the future demand for fiber services. Indeed, one can hardly predict what the actual fiber demand will be in the future while conceiving the network. Thus, many problem arises from this uncertainty. What if the network is not suited for the occurring demand scenario? If not, how much money one should spend to modify the network in order to supply that demand? What are the chances of having a "good" network regarding the demand? How can we maximize these chances? How to build a coherent model for that demand uncertainty?

These are the kind of difficult questions that often lead decision-makers to first build optimization models with a fixed, certain demand, as in [21, 34]. Indeed, by essence, the demand is highly volatile over space and time, then estimating it with accuracy can prove very hard in many cases. Moreover, in a competitive context as the telecommunications one, clients do not hesitate to switch their subscription contract from one operator to another, thus impacting the actual demand over time, even once the network is installed once and for all. Such imprecision could lead, in the case of network design problems, to two main problems:

- in case the demand exceeds the estimation, this demand will not be supplied by the network unless the operator does the necessary network extension, at a high cost, thus delaying the connection of the client to the network.

- in case the demand is below the estimation, it means the network has been over dimensioned, and it could be seen as a waste of time and money.

It is clear, by looking at these problems, that dealing with uncertainty will have a cost. That cost will be the amount of resources decision-makers are ready to invest to somehow protect themselves from what will happen later, and the cost he will spend later to repair, or adapt, his solution.

In this chapter, the objective is to propose a coherent modeling of the optimization problem that takes demand uncertainty into account. It will have to be coherent from a field point of view, regarding what is usually done in the literature. That means we want a model that tackles the real-life problem without limiting it to the realm of what is easily solvable. To that extent, various approaches will be discussed in order to make a modeling choice. The purpose of this chapter is to present the operational stakes linked to demand uncertainty, our comprehension processing of the problem, our modeling, our view of the approach and its motivation. The point is to clearly justify the trails we decided to investigate in order to tackle the generic issue of demand uncertainty. At the end of this chapter, we aim to obtain a comprehensive model of the problem we want to solve, so that the following chapters may deal with that solvability.

4.1 The choice of robust optimization

The generic framework under which we aim to deal with the demand uncertainty is robust optimization. This choice has been set as a rule before starting the present Ph.D, in order to restrict the exploration field. Indeed, other choices could have been made, like using stochastic optimization (or even inventing a whole new theory). However, this arbitrary choice, made for practical reasons at first, could have lacked relevance on a scientific point of view. Of course, this potential lack of relevance would have been detected during the bibliographical study (presented in section 1.2) and another research way would have been preferred to robust optimization. However, the initial choice of robust optimization was maintained since we estimated this approach to be the best, suited for the passive optical network deployment problem.

First, as mentioned in the bibliographical study, robust optimization is more focused on building solutions that are not too bad, whatever may happen. In opposition to this philosophy, stochastic optimization tries to get as many gain as possible from the

optimization, despite the data uncertainty, thus relying on its probabilistic description. Therefore, stochastic optimization will be suited for problems that are repeated over time, where one can afford to "loose" a little if it is assured to gain a lot globally. Problems like portfolio management are a good example of applied stochastic optimization.

Robust optimization is thus designed to deal with problems where feasibility is a key criterion, and where the decision-maker absolutely wants to avoid bad surprises regarding the cost of the solution. Also, it is a good alternative to stochastic optimization when a probabilistic description of the data uncertainty is not available, or hard to know with precision. For all these reasons, we believe that robust optimization was more suited to tackle the passive optical network design problem. Indeed, a network is deployed once and for all. Therefore, if a "bad" scenario occurs, there is no way to cheaply adapt the network to this scenario and in the end, the solution will be too much costly. We stress the fact that decision-makers are often risk averse, and they tend to prefer solution that will not be too bad whatever happens, with a good average cost performance. Moreover, as we will see later, the future fiber demand is very hard to quantify and estimate, and deriving predictive probability laws for that uncertain parameter would prove to be, in practice, irrelevant, if not wrong.

4.2 On the demand uncertainty in the passive optical network design context

In order to stay compliant with the notations throughout the document, let us denote by $\boldsymbol{d}$ the vector of uncertain demands. That vector has a size of $|V|$, V being the set of nodes. As it is stressed before, the customer demand is highly volatile over time and space, which makes it hard to estimate. But in robust optimization, there is no need for a precise description of the uncertainty, for it just needs to be contained in a bounded uncertainty set (see section 1.2.3). There are some sets that were proposed in the literature, they are reviewed in Section 1.2.3. Considering that d_i^{max} is the maximum number of client linked to the node $i \in V$, the only thing we can state for now is:

$$d_i \in \mathbb{N}, \;\; d_i \leq d_i^{max} \tag{4.1}$$

The first choice we made was to avoid ellipsoidal uncertainty sets like the one proposed by Ben-Tal and Nemirovski [7]. Indeed, there was no particular reason from the field to consider that kind of uncertainty set.

The second choice we made was to avoid the well-known Bertsimas and Sim uncertainty set. Let us recall that building this set requires to define, for each uncertain data,

a nominal value that would be its "average" value. Then the actual future value could be found "around" that central nominal value. That would mean we need to define that average value for every demand node, and the set of valid values for the future demand around this nominal value. Unfortunately, even this nominal value was not available in practice. The only thing that is known about the demand is that it will be between nothing and the total number of clients linked to the node. However, a way of circumventing this issue would be to notice that the worst case scenarios that we will try to optimize will always be scenarios where all uncertain data will be greater or equal than their nominal value. Therefore, a model of the uncertainty set that could fit our needs could be to consider the nominal value to be zero, the maximal variation around that nominal value to be d_i^{max} for all $i \in V$, and limiting the number of nodes having their maximal demand by a classic parameter Γ. But in a way, that would mean we would restrict ourselves to scenarios where some nodes are set their maximal values, and others to zero. And as we will see later, this is too restrictive in our problem.

Under these considerations, our final choice was to build an uncertainty set that is derived from what is expected by operational teams. Even if very little information is available about the demand behavior from the marketing department, they claim the most relevant parameter to consider for the passive optical network design problem under demand uncertainty is the maximum percentage of the total demand we expect to have in the end: this parameter can be seen as the maximum market share expected by the telecommunication operator for fiber subscriptions. This percentage will give us a total amount of demand $\bar{d}$ that will be distributed among all nodes of V in an unknown way. We believe that this uncertainty set, denoted by $\mathfrak{D}$, is much more natural to use in our context than the uncertainty set of Bertsimas and Sim. Therefore, we chose to use a polyhedral uncertainty set.

Definition 4.2.1. *The uncertain demand vector $\boldsymbol{d}$ is contained in the following uncertainty set $\mathfrak{D}$, defined from the parameter $\bar{d}$ that is the total amount of demand that will actually occur in the future.*

$$\mathfrak{D} = \left\{ \boldsymbol{d} \in \mathbb{N}^{|V|} \;\middle|\; d_i \leq d_i^{max}, \forall i \in V \;\; \texttt{and} \;\; \sum_{i \in V} d_i \leq \bar{d} \right\} \tag{4.2}$$

Remark 4.2.1. *In appearance, this approach seems to ignore the fact that urban areas themselves are changing over time. This means that new buildings may be built, thus implying more households linked to a node (or even new nodes with households to serve) to consider. It means that some actual d_i^{max} may be wrong in the future. However, stating a maximal demand d_i^{max} does not imply a loss of generality. Indeed, if a construction project is known around the node i so that there will be d_i' more potential households*

to serve, then one just need to add that number to d_i^{max}. If no construction project is known, yet it is highly possible that the urban areas will change in the following decades, telecommunication operators often add an over-dimensioning quantity to the local demand in order to be able to supply that potential future demand. It is thus always possible to increase d_i^{max} for all $i \in V$ so that every scenario is included in $\mathfrak{D}$.

Remark 4.2.2. *Note that if we consider the set $\mathfrak{D}$ with $\boldsymbol{d} \in \mathbb{R}^{|V|}$, the extremal points of $\mathfrak{D}$ would be integer points.*

Remark 4.2.3. *The model proposed for $\mathfrak{D}$ states that we will know the demand limitation over the whole area V. However, it is also possible that for some sub areas $S^V \subset V$, deeper information is available. In that case, adding such information to the set $\mathfrak{D}$ would be useful to enrich its description. We denote by $\mathfrak{D}'$ such an uncertainty set.*

$$\mathfrak{D}' = \left\{ \boldsymbol{d} \in \mathbb{N}^{|V|} \;\middle|\; d_i \leq d_i^{max}, \forall i \in V \;\; \texttt{and} \;\; \begin{array}{l} \sum_{i \in V} d_i \leq \bar{d} \\ \sum_{i \in S^V} d_i \leq \bar{d}_S, \forall S^V \in V \end{array} \right\} \tag{4.3}$$

Most of the methods presented in the following chapters will also be compliant with the uncertainty set $\mathfrak{D}'$. For the sake of clarity, results will not be extended to that case every time.

4.3 On the field deployment practices by operational teams

Driven by the objective of proposing an approach that can answer real operational concerns, let us quickly review our observations and feedback we obtained from field deployment operators. Indeed, engineering rules changed over the years, along with deployment processes. They will be detailed below and every modeling choice of the chapter will be made so as to be as close as possible from them. Some may seem contradictory with what was proposed in Chapters 1, 2 and 3. These differences are chronologically explained since our work on the robust approach started last. Therefore, deployment hypothesis that were valid at the time had changed. Note that the deployment rules we are going to detail here are those used by Orange. For that reason, we will only give a generic description of it, so it is useful for our problem's modeling. It is of course possible that other operators decided to do otherwise.

The first important change is that in practice, the only relevant decision to make is the placement of only one splitter level. Indeed, even if all deployed architectures are still having 2 or 3 splitter levels, Orange decided that for the sake of simplicity, it would

be best to place some of these splitters in advance. Note that this in the line with what was proposed in Chapter 2 with the splitter delocation rule (see Section 2.1.1 for specific details). In practice, the last splitter level is now put directly on the client's node, so as to supply all the demand. In the case of a 3 level architecture, the first splitter level is put at the OLT. In both cases, only one splitter level remains to be placed and optimized.

The second important change is about duct capacities. In practice, they are simply ignored. There are two main reasons for that: the fact that most of the time the remaining capacity is very large and the technological advancement which makes fiber cables getting much thinner that before. Therefore, the chances that a deployment solution is unfeasible due to capacity constraints is extremely low. Moreover, before starting the deployment of a chosen solution, operational entities are always going on the field before to check if the remaining capacity is large enough for the solution to be deployed. In case there is a duct that can not be used as planned (due to an error or an omission in the database), the solution is changed accordingly. This case rarely occurs on the field, this is why operational entities prefer not to consider capacities in the first place and then changing the solution before making the deployment if a limiting capacity is found on the field.

The last point is the most important since it deals with demand uncertainty. Indeed, operators are fully aware of the difficulty of building the network without knowing the demand precisely. To that extent, operational entities decided to build the network sequentially. The first step is to draw fibers from the OLT to every potential household in the area. Indeed, even if we cannot be assured that they will subscribe our fiber offer, it is much simpler to give every client an incoming fiber than coming back later to add some if the demand changes. As fibers are drawn, rooms are opened to they can receive optical splitters, to sustain fiber paths. However, as the branching in and out of a splitter is easy to perform, and most of all easy to modify, it was thus decided that optical splitters would be the key decision that will adapt the network to the actual demand. In practice, optical splitters are thus placed accordingly to the demand. As in the first step, optical fibers were deployed for everyone, it is always possible to find a splitter configuration that can fit any demand. Of course, depending on the demand, more or less splitters will be needed. This is why the robust approach has been chosen.

4.4 Modeling the problem as a single stage robust optimization problem

Regarding the bibliography in section 1.2, the first approaches that were designed for robust optimization were single stage approaches. However, as seen in section 1.2.5, it is not suited for every problems. Thus, at this point, the reader may already know that a

single stage approach for this problem will quickly reach its limits due to the specificities of Right-Hand-Side uncertainty. However, this raises the opportunity to discuss this issue for our problem, and to highlight some of its particularities.

4.4.1 A formulation based on arc variables

For the deterministic version of the problem

For the time being, let us propose a model for a deterministic version of the problem (that is, without uncertainty) which is based on the formulation proposed by Trampont et al. in [50, 21], introduced in section 1.1.2 and using the same notations.

$$(PON^{arc}_{det}) \begin{cases} \min\limits_{\mathbf{f},\mathbf{z}} & \sum\limits_{i=1}^{n} C\mathbf{z}_i + \sum\limits_{(i,j)\in E}\sum\limits_{k=1}^{2} c^k_{ij}\left(\mathbf{f}^k_{ij} + \mathbf{f}^k_{ji}\right) \\ \text{s.t.} & \sum\limits_{j\neq i} \mathbf{f}^1_{ji} \geq \mathbf{z}_i + \sum\limits_{j\neq i} \mathbf{f}^1_{ij},\ \forall i \in V^* & (4.4) \\ & \sum\limits_{j\neq i} \mathbf{f}^2_{ji} + m\mathbf{z}_i \geq d_i + \sum\limits_{j\neq i} \mathbf{f}^2_{ij},\ \forall i \in V & (4.5) \\ & \mathbf{z}_i \in \mathbb{N},\ \ \forall i \in V^*, \mathbf{z}_0 = 0 \\ & \mathbf{f}^k_{ij} \in \mathbb{R},\ \ \forall (i,j) \in E, k = 1,2 \end{cases}$$

Here, several differences (from the original deterministic model presented in Section 1.1.2 and studied in [50, 21]) shall be highlighted. First, constraints (4.4) and (4.5) are set to be inequalities instead of equalities. Both formulations are equivalent, however, in a robustness context, as shown in [2, 43] for example, dealing with equality constraints can prove less relevant. Indeed, once demand is uncertain, the incoming number of fibers on a node may not be set too strongly, as one may want to keep several branching options for satisfying the demand in different ways. Moreover, we are only considering a single level splitter architecture and we neglect capacity constraints to be consistent with assumptions of Section 4.3.
Note that the integrity of variables $\mathbf{f}$ can now be relaxed since capacities have been removed (see remark 1.1.4).

For the single stage robust version of the problem

Problem (PON^{arc}_{det}) being defined, let us examine the demand uncertainty in this problem. The uncertain data here are the right-hand-side terms $d_i, \forall i \in V$. Let us use the uncertainty set $\mathfrak{D}$ defined in the previous section 4.2 by the equation (4.2) since it is more suited to our modeling needs. Let us define the single stage robust counterpart problem of

(PON^{arc}_{det}), denoted by (PON^{arc}_{rob}), which is the problem consisting in finding the cheapest network so that any demand can be satisfied. Note that for now, the sequential building of the network presented in Section 4.3 is not taken into account. It can be modeled as follows:

$$(PON^{arc}_{rob})' \begin{cases} \min\limits_{\mathbf{f},\mathbf{z}} \sum\limits_{i=1}^{n} C\mathbf{z}_i + \sum\limits_{(i,j)\in E} \sum\limits_{k=1}^{2} c^k_{ij} \left(\mathbf{f}^k_{ij} + \mathbf{f}^k_{ji}\right) \\ \text{s.t.} \sum\limits_{j\neq i} \mathbf{f}^1_{ji} \geq \mathbf{z}_i + \sum\limits_{j\neq i} \mathbf{f}^1_{ij}, \ \forall i \in V^* \\ \quad \sum\limits_{j\neq i} \mathbf{f}^2_{ji} + m\mathbf{z}_i \geq \hat{d}_i + \sum\limits_{j\neq i} \mathbf{f}^2_{ij}, \ \forall i \in V \\ \quad \mathbf{z}_i \in \mathbb{N}, \ \ \forall i \in V^*, \mathbf{z}_0 = 0 \\ \quad \mathbf{f}^k_{ij} \in \mathbb{R}, \ \ \forall (i,j) \in E, k = 1,2 \end{cases}$$

with $\hat{d}_i = \min\{d^{max}_i, \bar{d}\}$.

Remark 4.4.1. *Note that $\hat{d}_i = \min\{d^{max}_i, \bar{d}\}$. Thus if, for all nodes i, $d^{max}_i \leq \bar{d}$, then (PON^{arc}_{rob}) is exactly the same as (PON^{arc}_{det}) since $\hat{d}_i = d^{max}_i$, $\forall i \in V^*$. In the following, we will always consider that $d^{max}_i \leq \bar{d}$, $\forall i \in V^*$ because it never happens in reality.*

Thus, this robust single stage problem reduces to the deterministic passive optical network design problem. Indeed, supplying the maximal demand possible on every node is the best way to ensure a complete feasibility. Moreover, as it is discussed in the following section, using d^{max}_i for all nodes (instead of $\bar{d}$ if $d^{max}_i > \bar{d}$) is more relevant from an operational field point of view. However, knowing the fact that not every household in the area will contract a fiber subscription in the end, we can reasonably state that the network is over dimensioned, especially in terms of optical splitters. This issue is specific to right-hand-side uncertainty formulations, as shown in section 1.2.5, for they often lead to trivial single stage formulations where the limitations put on the uncertainty set $\mathfrak{D}$ are of little effect on the optimization. Moreover, as it is recalled in this section by Minoux [38], dualizing (PON^{arc}_{rob}) in order to have uncertain values in a row-wise configuration is out of the question since it is a completely different problem: the dual of the robust counterpart is different from the robust counterpart of the dual. Therefore, unless having another formulation, solving the problem with a single stage approach present little interest, both in practice and in theory.

4.4.2 A formulation based on path variables

For the deterministic version of the problem

In their work, Kim et al. [34] proposed a path formulation for solving the deterministic version of the problem with cabling constraints in an arborescence. That model is also used in Chapter 3. What is interesting us here is that in their model, demands were introduced in a row-wise way in the constraint matrix. We will derive the modeling principles they use to propose a single stage robust problem that may perform better than the one presented in the latter section. However, since we remain in a single-stage context, we cannot include the sequential building of the network explained in Section 4.3.

Let us denote by c_{ij}^k, $k = 1, 2$, the cost of routing one level k fiber from node i to node j in G, along the shortest path. Using only the shortest path is possible since we do not consider capacity constraints in this problem. C_i is now the cost of installing a splitter at node i. This splitter cost is a combination of the cost of the splitter itself C and the cost of routing a level one fiber from the OLT to the node i along the shortest path. Then $C_i = C + c_{0i}^1$. All other data notations remains the same.

Splitter variables are still denoted by $\mathbf{z}_i$ for all $i \in V^*$. Let us denote by $\mathbf{x}_{ij}$ the continuous variable that is the proportion (between 0 and 1) of the demand at node j that is served by splitters located at node i along the shortest path in G from i to j. It is important to note that we relax integrity constraints on the fiber variables and that, as we will see later, it may result in non integer number of fibers, even though there are no capacity constraints.

Thus, the deterministic passive optical network design problem can have the following path formulation:

$$(PON_{det}^{path}) \begin{cases} \min\limits_{\mathbf{f},\mathbf{z}} & \sum\limits_{i=1}^{n} C_i \mathbf{z}_i + \sum\limits_{(i,j)\in V^2} c_{ij}^2 d_j \mathbf{x}_{ij} \\ \text{s.t.} & \sum\limits_{i\in V^*} \mathbf{x}_{ij} = 1, \, \forall j \in V^* & (4.6) \\ & \sum\limits_{j\in V^*} d_j \mathbf{x}_{ij} \leq m\mathbf{z}_i, \, \forall i \in V^* & (4.7) \\ & \mathbf{z}_i \in \mathbb{N}, \;\; \forall i \in V^*, \\ & \mathbf{x}_{ij} \in \mathbb{R}_+, \;\; \forall (i,j) \in V^{*2} \end{cases}$$

The objective function minimizes the cost of installing a splitter at a given node plus the cost of routing fibers in the graph. Constraints (4.6) ensure that the demand satisfaction at node $j \in V^*$ is entirely served for all j. Constraints (4.7) ensure that there are enough optical splitters located at node $i \in V^*$ to produce the outgoing fiber flow

from node i towards all nodes of the graph (including i). Note that variables $\mathbf{x}_{ij}$, even if they are defined in $\mathbb{R}_+$, can not exceed 1.

For the single stage robust version of the problem

Let us now consider that the demand vector $\boldsymbol{d}$ is uncertain. From the literature (see Section 1.2.4), we know that there are efficient methods to take row-wise uncertainty into account. That consists in protecting every constraint against uncertainty by integrating an optimization problem within each constraint. However, before applying this approach, we must introduce the uncertainty in our problem appropriately. The idea is to consider a "relaxed" version of the column-wise uncertainty defined in 4.2.

In (PON^{path}_{det}), there are d_i factors in the objective function and in constraints (4.7). But before applying the results from the literature, let us discuss the purpose of robustly optimizing our problem. As mentioned in the latter section 4.4.1, being feasible to every scenario is actually very easy, since deploying the network for everyone is already, in terms of feasibility, the "most" robust solution. Thus it is important to precise that the point of dealing with uncertainty, in our operational context, is to reduce the total cost, knowing that the full demand scenario will never occur, while ensuring feasibility.

Our problem has 2 types of variables: fibers and splitters. From the fiber point of view, one can easily see that whatever our robust approach is, we will have to supply every clients with fibers, as it is done on the filed (see Section 4.3). It implies that on every demand node i, the number of coming last level fibers shall always be equal (or greater) to d_i^{max}. That fact holds even when $\bar{d} \leq d_i^{max}$ (which is very unlikely in practice) since within a building, one can hardly know which client will chose to be connected to the network. In other words, the only way to ensure feasibility from a fiber point of view is to provide everyone in the area with a last level fiber. This is also coherent with field requirements since when networks are deployed, fiber cables are pulled into ducts and clients are connected once and for all. It would not be acceptable to come back once the network has been deployed long ago in order to add fibers so that a non connected client that desires to join can do so. Therefore, our margin of action is in optical splitters. While fibers ensure a potential connexion to the OLT, the key item that make the link real is the optical splitter. In practice, splitter branching can be changed according to the actual demand and operators already exploit that possibility to reduce the number of splitters they will install in the network. Since in practice, optical splitters tends to be much more expensive than fibers, one can easily see that reducing their number can result in a non negligible gain, and this is why we tackled uncertainty in this study.

Once these considerations are taken into account, let us take our path formulation introduced in this section. We propose the following single stage robust problem for the passive optical network design problem:

$$(PON^{path}_{rob}) \begin{cases} \min\limits_{\mathbf{x},\mathbf{z}} \sum\limits_{i=1}^{n} C_i \mathbf{z}_i + \sum\limits_{(i,j)\in V^2} c_{ij}^2 d_j^{max} \mathbf{x}_{ij} \\ \text{s.t.} \sum\limits_{i\in V^*} \mathbf{x}_{ij} = 1, \ \forall j \in V^* \\ \quad \sum\limits_{j\in V^*} d_j \mathbf{x}_{ij} \leq m\mathbf{z}_i, \ \forall i \in V^*, \forall d \in \mathfrak{D} \\ \quad \mathbf{z}_i \in \mathbb{N}, \ \ \forall i \in V^* \\ \quad \mathbf{x}_{ij} \in \mathbb{R}_+, \ \ \forall (i,j) \in V^{*2} \end{cases}$$

This formulation is a possible single stage robust version of the deterministic problem (PON^{path}_{det}) where we consider that fibers are deployed for every one, which is why we consider the cost of fiber as if they were deployed for the maximal demand on every node. There is one important thing to stress here, in order to clearly understand what this model does, and more importantly, what it does not. Fiber routing variables $\mathbf{x}_{ij}$ denote a proportion of the total fiber flow that supplies the demand node j. In the deterministic version of the problem, it is seen as a way of rewriting explicit fibers variables. But in our robust context where the demand is unknown, it yields another decision, which is more than just routing fibers. Indeed, $\mathbf{x}$ variables also indicate how the demand will be served, that is, what fibers are we going to light in order to match the demand. This is more than just routing fibers, it also decides how fibers will be used. The choice of variables $\mathbf{x}$ thus imposes a rigid demand supplying strategy. Of course, this could be seen as restrictive since in practice, operators will probably optimize the use of their installed fibers, and there is no evidence that choosing a routing strategy that uses the same proportion of fibers on every incoming path is good, or even relevant. But being able to optimize the fiber branching once the demand is revealed and take it into account in our optimization approach is not possible in a single stage robust context, since all decisions must be taken before the uncertainty is revealed and this includes the branching strategy. At last, note that we do not consider the uncertainty globally, but only constraint by constraint. Indeed, for each single demand satisfaction constraint, we consider uncertainty locally, independently of other demand satisfaction constraints.

Figure 4.1 shows an example of this principle. On the left side, a feasible solution of the problem (PON^{path}_{rob}) is presented for the maximum demand (note that this solution may not be optimal). On the right side, for two given demands (15 and 40), it shows how, in our single stage robust problem, fibers will implicitly be used in order to supply the demand. For the upper figure, one could imagine another way of supplying the demand

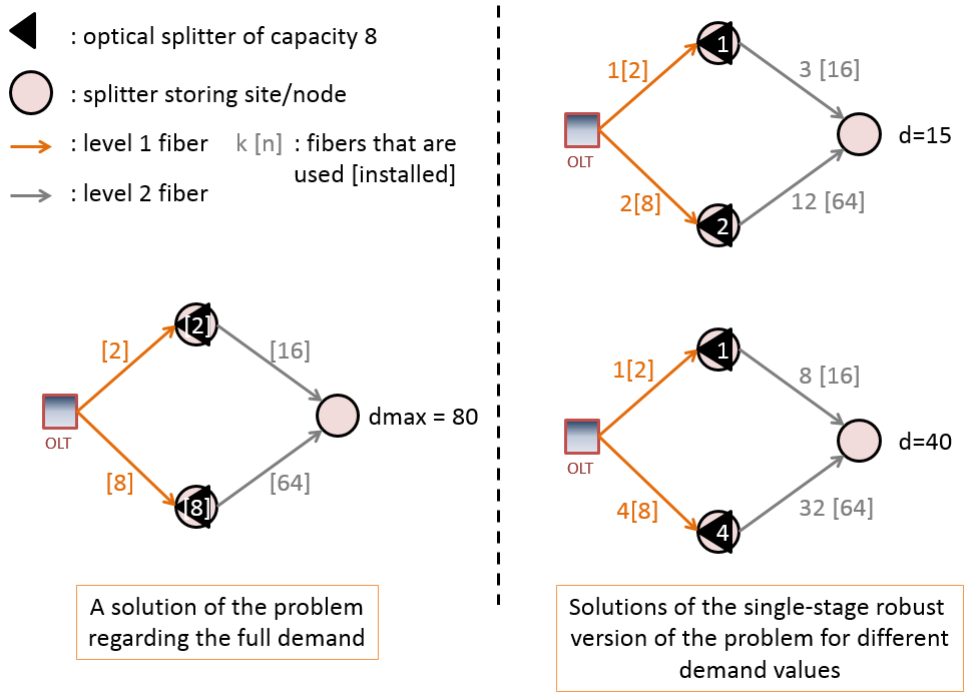

Figure 4.1: An illustration of how the chosen routing strategy based on the use of "proportional" installed fiber flow influences solutions

by , for example, using 15 fibers coming from the upper node and none from the lower node. This choice would be optimal in terms of splitters since it would only need 2 to supply the demand. But our model cannot include such an adaptability to the demand as the best routing strategy will not be the same depending on the configuration.

Remark 4.4.2. *Note that the routing imposed by the proportional routing strategy may not correspond to an integer number of fibers. Actually, most of the time, it will not. Demand values showed on the right side of figure 4.1 were well chosen. Setting $\bar{d}$ to 17 or 53 would lead to non integer fiber flows. However, this is due to its restrictive nature in this context, and is one of the many side effects of this approximation.*

However, even with this restrictive modeling, let us show how it could help reducing

the cost of the network by rewriting the problem.

$$(PON^{path}_{rob}) \begin{cases} \min\limits_{\mathbf{x},\mathbf{z}} & \sum\limits_{i=1}^{n} C_i \mathbf{z}_i + \sum\limits_{(i,j)\in V^2} c_{ij}^2 d_j^{max} \mathbf{x}_{ij} \\ \text{s.t.} & \sum\limits_{i\in V^*} \mathbf{x}_{ij} = 1,\ \forall j \in V^* & (4.8) \\ & \sum\limits_{j\in V^*} d_j \mathbf{x}_{ij} \leq m\mathbf{z}_i,\ \forall i \in V^*, \forall d \in \mathfrak{D} & (4.9) \\ & \mathbf{z}_i \in \mathbb{N},\ \ \forall i \in V^* & (4.10) \\ & \mathbf{x}_{ij} \in \mathbb{R}_+,\ \ \forall (i,j) \in V^{*2} \end{cases}$$

Constraints (4.9) must be verified for all possible realizations of $d \in \mathfrak{D}$, which leads to a huge sized model.

Remark 4.4.3. *The optimal value of the problem* (PON^{path}_{rob}) *is lower or equal than the optimal value of the problem* (PON^{path}_{det})*.*

Proof. Constraints (4.9) of problem (PON^{path}_{rob}) can be rewritten as such:

$$\max_{d\in\mathfrak{D}} \left(\sum_{j\in V^*} d_j \mathbf{x}_{ij} \right) \leq m\mathbf{z}_i,\ \forall i \in V^*$$

By definition of $\mathfrak{D}$ (as $\mathbf{x}$ variables are all positive), we have

$$\max_{d\in\mathfrak{D}} \left(\sum_{j\in V^*} d_j \mathbf{x}_{ij} \right) \leq \sum_{j\in V^*} d_j^{max} \mathbf{x}_{ij}$$

Therefore, each robust splitter satisfaction constraint is either the same or less restrictive than their deterministic counterpart. □

The proof of Proposition 4.4.3 shows how the robust approach has a chance of reducing the number of installed splitters, therefore the total cost of the problem. The following theorem shows how to solve it.

Theorem 4.4.1. (PON^{path}_{rob}) *can be rewritten as the following mixed integer linear pro-*

gram:

$$(PON^{path}_{rob})\begin{cases} \min\limits_{\mathbf{x},\mathbf{z}} & \sum\limits_{i=1}^{n} C_i\mathbf{z}_i + \sum\limits_{(i,j)\in V^2} c_{ij}^2 d_j^{max}\mathbf{x}_{ij} \\ s.t. & \sum\limits_{i\in V^*}\mathbf{x}_{ij} = 1,\ \forall j \in V^* & (4.11) \\ & \sum\limits_{j\in V^*} d_j^{max}\omega_j^i + \bar{d}\mathbf{\Omega}_i \leq m\mathbf{z}_i,\ \forall i \in V^* & (4.12) \\ & \mathbf{\Omega}_i + \omega_j^i \geq \mathbf{x}_{ij},\ \forall (i,j) \in V^{*2} & (4.13) \\ & \mathbf{z}_i \in \mathbb{N},\ \ \forall i \in V^*, \\ & \mathbf{x}_{ij} \in \mathbb{R}_+,\ \ \forall (i,j) \in V^{*2} \\ & \omega_{ij} \in \mathbb{R}_+,\ \ \forall (i,j) \in V^{*2} \\ & \mathbf{\Omega}_i \in \mathbb{R}_+,\ \ \forall i \in V^* \end{cases}$$

Proof. The problem (PON^{path}_{rob}) can first be rewritten as:

$$(PON^{path}_{rob})\begin{cases} \min\limits_{\mathbf{x},\mathbf{z}} \sum\limits_{i=1}^{n} C_i\mathbf{z}_i + \sum\limits_{(i,j)\in V^2} c_{ij}^2 d_j^{max}\mathbf{x}_{ij} \\ \text{s.t.} \sum\limits_{i\in V^*}\mathbf{x}_{ij} = 1,\ \forall j \in V^* \\ \max\limits_{d\in\mathfrak{D}} \left(\sum\limits_{j\in V^*} d_j\mathbf{x}_{ij} \right) \leq m\mathbf{z}_i,\ \forall i \in V^* \\ \mathbf{z}_i \in \mathbb{N},\ \ \forall i \in V^* \\ \mathbf{x}_{ij} \in \mathbb{R}_+,\ \ \forall (i,j) \in V^{*2} \end{cases}$$

For a given $i' \in V^*$, let us write the inner maximization problem in d, denoted by $R_{\mathfrak{D}}(\mathbf{x})$ contained in the splitter satisfaction constraint for node i':

$$R^{ii'}_{\mathfrak{D}}(\mathbf{x})\begin{cases} \max\limits_{d} \sum\limits_{j\in V^*} d_j\mathbf{x}_{ij} \\ \text{s.t. } d_j \leq d_j^{max},\ \forall j \in V^* & (\omega_j^{ii'}) \\ \quad \sum\limits_{j\in V^*} d_j \leq \bar{d} & (\Omega_{ii'}) \\ \quad d_j \in \mathbb{R}_+,\ \forall j \in V^* \end{cases}$$

Here, d_j variables have their integrity relaxed. Indeed, Remark 4.2.2 shows that extremal points of $\mathfrak{D}$ are integer. That inner maximization problem is a continuous linear program that can be dualized, which gives:

$$DR^{ii'}_{\mathfrak{D}}(\mathbf{x})\begin{cases} \min\limits_{\omega^{i'},\mathbf{\Omega}_{i'}} \sum\limits_{j\in V^*} d_j^{max}\omega_j^{i'} + \bar{d}\mathbf{\Omega}_{i'} \\ \quad \text{s.t.} \mathbf{\Omega}_{i'} + \omega_j^{i'} \geq \mathbf{x}_{i'j},\ \forall j \in V^* \end{cases}$$

One can replace the maximization problem by its dual counterpart. Any feasible values taken by $\sum_{j \in V^*} d_j^{max} \omega_j^{i'} + \bar{d} \mathbf{\Omega}_{i'}$ will thus be greater or equal to $\max_{d \in \mathfrak{D}} \left(\sum_{j \in V^*} d_j \mathbf{x}_{ij} \right)$, making the minimization useless as the global optimization problem will tend to minimize the dual anyway. □

Remark 4.4.4. *Denoting by n_d the number of demand nodes (that is: $n_d = |V^*|$), the mixed integer program has $n_d^2 + n_d$ additional continuous variables and n_d^2 additional constraints compared to the path formulation of the deterministic problem.*

Experimental study of the gain induced by the single stage robust problem with a path based formulation

Now that theorem 4.4.1 gives us a formulation that can be solved in practice, we aim to conduct a quick empirical study on the potential gains offered by the single stage robust approach. Since we already know that our single stage model has limitations, we intend to propose more complex models. But for now, our objective is to assess the relevance of our scientific approach. Therefore, we will not put many efforts in improving the solving of the single stage problem as we strive for both qualitative and quantitative information on what makes our problem robust so that we can decide our next course of action.

The tests presented in this section will not appear in their exhaustive form, since we just aim to illustrate a behaviour that was always observed in the tests we performed. On several instances between 10 and 50 nodes, randomly generated as described in section 6.2.4, both problems (PON_{det}^{path}) and (PON_{rob}^{path}) were solved. For (PON_{rob}^{path}), we chose 3 possible values for $\bar{d}$ in order to highlight the impact of the uncertainty set's size:

- $\bar{d}_{25\%} = \left\lceil 0.25 \times \sum_{i \in V^*} d_i^{max} \right\rceil$
- $\bar{d}_{50\%} = \left\lceil 0.50 \times \sum_{i \in V^*} d_i^{max} \right\rceil$
- $\bar{d}_{75\%} = \left\lceil 0.75 \times \sum_{i \in V^*} d_i^{max} \right\rceil$

Moreover, we set the optical splitter cost to 1000, and we let the fiber cost per distance unit increase from 0 to 10. For each cost configuration and each market share $\bar{d}_{\alpha\%}$, we compute the gain due to the robust problem given by $\frac{v(PON_{det}^{path}) - v(PON_{rob}^{path})}{v(PON_{det}^{path})}$.

Figure 4.2 gives results of these tests for a given instance of 30 nodes where each curve corresponds to a given α value. The gain due to the robust approach is reported with

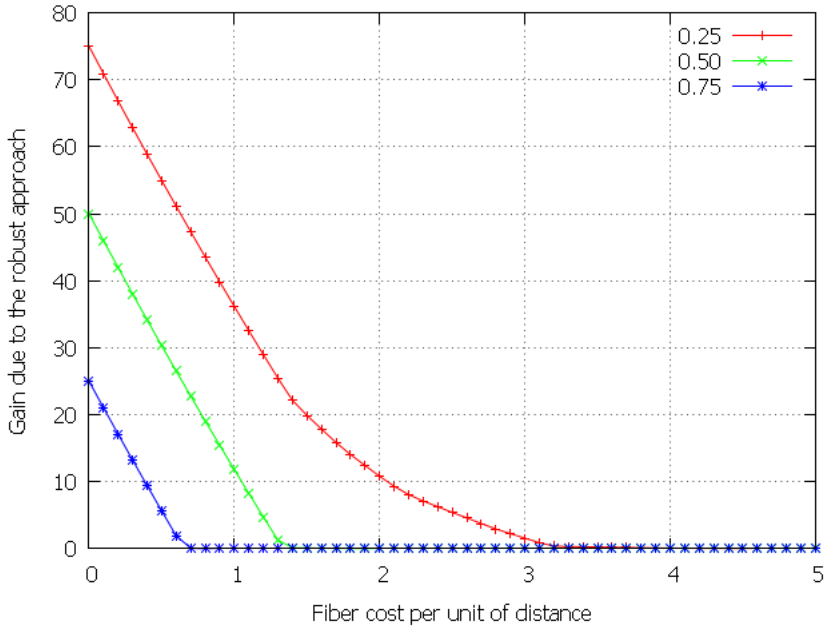

Figure 4.2: An illustration of the gain due to the robust approach depending on the fiber cost for 3 different values of $\bar{d}$

respect to the fiber cost increase. The first observation is that the smaller $\bar{d}_{\alpha\%}$ is, the higher is the gain. Indeed, when fibers cost nothing, we observe that the gain is exactly $100\%-\alpha\%$. This is possible by putting splitters on a single node that supply every demand nodes, regardless of the fiber routing since they cost nothing. The second observation is that as the fiber cost increases, the gain tends to diminish quite fast. Indeed, if fibers cost something, then their routing is not free any more, and the optimization thus finds a good trade-off between fibers and splitters. No further gain is observed after a while, and the solution proposed by the single stage robust problem is exactly the same as the one proposed by its deterministic counterpart.

Remark 4.4.5. *Let us consider the inner maximization problem $R_{\mathfrak{D}}(\mathbf{x})$ that is embedded in the splitter satisfaction constraint of problem (PON^{path}_{rob}) for a given $\mathbf{x}'$ and i':*

$$R_{\mathfrak{D}}(\mathbf{x}) \begin{cases} \max\limits_{d} \sum\limits_{j \in V^*} d_j \mathbf{x}'_{i'j} \\ s.t.\ d_j \leq d_j^{max},\ \forall j \in V^* \\ \quad \sum\limits_{j \in V^*} d_j \leq \bar{d} \\ \quad d_j \in \mathbb{R}_+,\ \forall j \in V^* \end{cases}$$

The solution of this easy (in terms of complexity) problem is immediate. Considering we have a sort of "demand budget" $\bar{d}$ to spend, we will assign demand to the demand node

j that have the highest strictly positive $\mathbf{x}'_{i'j}$ *value, up to* d_j^{max}*. Once the node is full, we just have to iterate the process by taking nodes according to* $\mathbf{x}'_{i'j}$ *values in a decreasing order. We stop when there is no demand budget any more or when there is no other remaining strictly positive* $\mathbf{x}'_{i'j}$*. Note that if we stop before consuming all the demand budget, the solution of* $R_{\mathfrak{D}}(\mathbf{x})$ *is exactly the same as in Soyster's approach. Therefore, to make use of the demand limitation* $\bar{d}$ *in order to observe a gain, the node* i' *needs to supply enough demand nodes so that the total demand it reaches exceeds* $\bar{d}$*. Depending on* $\bar{d}$*, this means that only a small number of nodes will be used for splitters, thus making the global length of deployed optical fibers increase. Consequently, especially on instances where both demands and distances are "high", the gain due to splitter centralization may not compensate the loss in terms of optical fibers. Hence, in addition to the branching strategy limitations, this is the major reason for the sharp decrease of the gain observed in Figure 4.2 when the fiber cost increases.*

We can draw several conclusions from that study. The first one is that robust optimization does present a practical interest for our problem. Indeed, depending on the cost structure of the problem, the single stage approach already allows us to find substantial gains by exploiting the fact that the overall demand will be limited. However, and that is our second conclusion, that cost high dependency is the main drawback on the single stage approach, since its lack of adaptability to the uncertainty makes it efficient for a very limited range of cost values. Indeed, we do not fully consider the operational process that consists in installing optical splitters in the future, once the demand is known.

Regarding these conclusions, it seems that robust optimization is a pertinent scientific choice, but our model needs to include adaptability to uncertainty. That is why our next course of action is to tackle the passive optical network design problem by the means of two-stage robust optimization.

4.5 Modeling the problem as a two stage robust optimization problem

While exploring single stage robust approaches, we already mentioned that on the field (see Section 4.3), telecommunication operators are actually adapting their network to the outcome demand. For example, it can be done by deciding which fiber will supply which client, with which optical splitter. Therefore, they do not apply the trivial super robust solution that consists in deploying a network as if the maximal total demand was met everywhere. However, our single stage approaches failed to take that field adaptability into account. We thus aim to propose two stage approaches for the design of our network, so that the total cost of the network can be more reduced.

4.5.1 The general Modeling of the two stage robust passive optical network design problem

The concept of multi-stage optimization is not specific to robust optimization, since it is already used in stochastic optimization for example [18]. The main principle is that decisions are not taken all at once, but some of them can be delayed in the future. In our context, we will consider a two-stage optimization problem, where some of our decisions may be taken once the demand is revealed. For this first study of the problem, we will consider that the demand is revealed all at once. Of course, in practice, all clients will not purchase the fiber connexion offer at the same time, and we could imagine a problem where the total demand increases gradually, following several time horizons, thus exploiting it to reduce the cost over time. But as mentioned before, the customer demand is highly unpredictable, therefore, knowing what time horizons would be pertinent is, by itself, a difficult task in this context. Consequently, we choose to deal with the cost of the network once the final total demand is revealed. In addition, the fact that the demand may change over time will still be included in our robust solutions since we are dealing with a huge set of solutions that includes all possible scenarios within the limit of $\bar{d}$.

But before introducing the model, it is important to define with accuracy what our decisions are. Indeed, decision variables are, in their own way, actual decisions. But the mathematical formalism often leads to aggregate information, that is to say various decisions, into a single variable. For example, splitter variables $\mathbf{z}$ actually aggregate two different decisions: where splitters are located and how many of them are installed. Therefore it is crucial to identify the elementary decisions that one has to take while deploying a passive optical network, independently of the mathematical formalism. We already applied that kind of reasoning while creating the model for the single stage robust problem (PON^{path}_{rob}) presented in the latter section. The important thing to stress is that one can not just pick a deterministic problem, decide which variables will be first stage variables and which will be recourse variables and solve the whole problem as it is. The decisions that we identified are:

- decisions related to optical splitters:
  - the sites where optical splitters will be located (1s)
  - the number of optical splitters to be installed (2s)
  - the branching of optical splitters to fibers in order to supply the customer demand (3s)
- decisions related to optical fibers:

- the point of departure and arrival of each optical fiber (1f)
- the number of optical fibers to deploy (2f)
- the routing path for each optical fibers to reach their arrival point (3f)
- the optical fiber allocation to the customer demand (which fibers will be used for customer demand supplying) (4f)

In the deterministic models we used so far, all these decisions are resumed by only 2 sets of variables: fibers and splitters. But in a two-stage robust context, we need to decide which decisions can be delayed. As mentioned before, operators insist on bringing one fiber to each potential customer, even though it is known that most of it will never be used. Therefore, the only gain margin is to be found in optical splitters. Indeed, while dealing with the deployment of the passive optical network, operators do not have to install optical splitters immediately, as long as their potential localization has already been decided. Thus, as customers are eventually requesting to be connected, the operator just has to install the required number of splitters amongst the pre decided splitter locations, branching the right fibers in the right slots, and so on. Note that the branching can be modified if needed. Of course, there can be many options for installing and branching splitters and that is indeed an optimization problem to do so while trying to minimize the total amount of installed splitters.

As the reader may anticipate, the choice of the mathematical formalism will not be trivial and we shall discuss it in the next section. For now, we need to decide which decisions should be taken here-and-now, and which one can be delayed. Using our decision decomposition presented above, we decided that decisions that must be taken before revealing the uncertainty are (1s), (1f), (2f) and (3f). Hence, the decisions we allow to delay in the future are (2s), (3s) and (4f). Let us denote by $\mathbf{h}_1$ (respectively $\mathbf{h}_2(d, \mathbf{h}_1)$) the set of first stage (respectively recourse, thus depending on the uncertain vector d) variables that do represent the selected decisions (1s), (1f), (2f) and (3f) (respectively (2s), (3s) and (4f)). The associated set of possible solutions is denoted by H_1 (respectively $H_2(d, \mathbf{h}_1)$). The associated cost function is $C_1(\mathbf{h}_1)$ which denotes the cost of installing optical fibers only (respectively $C_2(d, \mathbf{h}_1, \mathbf{h}_2)$ which denotes the cost of installing optical splitters in order to supply the customer demand). The two stage robust passive optical network design problem (PON_{rob}^{2stage}) can thus be modelled as such:

$$(PON_{rob}^{2stage}) = \min_{\mathbf{h}_1 \in H_1} \left[C_1(\mathbf{h}_1) + \max_{d \in \mathfrak{D}} \left(\min_{\mathbf{h}_2(d,\mathbf{h}_1) \in H_2(d,\mathbf{h}_1)} C_2(d, \mathbf{h}_1, \mathbf{h}_2) \right) \right] \tag{4.14}$$

This problem is, as we justified before, the one that best fits with what is done in reality. However, it is possible to think of other robust problems which may be more

suited depending on the context and deployment policy, that can be seen as variants from the one presented above, depending on the repartition of elementary decisions in the time-line. Remark 4.5.1 presents a possible variant of that problem that is, in our opinion, the most relevant after ours. However, we now make the choice to stick to (PON_{rob}^{2stage}) in the following of our study since, as it will be presented later, this problem is also much harder to solve that its variant, thus making it more challenging and, in a way, interesting.

Remark 4.5.1. *An interesting variant, from an industrial point of view, denoted by* ($PON_{rob}^{2stage\prime}$)*, consists in taking decision (2s) during the first stage, instead of the second. This means that in practice, optical splitters will be installed before knowing the actual demand, but their branching can still be delayed. This way, the only human intervention needed is to perform that branching. This problem can be seen as the direct two-stage version of the problem* (PON_{rob}^{path})*, where the branching strategy is made free from the restrictive proportional rule imposed in the single stage problem. Note that in this problem, the recourse problem does not yield any cost, since the whole network is deployed before knowing the uncertainty. Therefore, the recourse problem is actually a decision problem (is there a demand scenario that is infeasible for this network?) instead of an optimization problem. That aspect shall be explained later.*

4.5.2 The choice of a formulation for (PON_{rob}^{2stage})

In order to have a fitting formulation for our problem, we need to properly define our variables and constraints. Of course, as the passive optical network design problem has already been studied, there are existing models that are presented in the bibliographical study in section 1.1. Thus, Modeling itself is not an issue, but the choice of the Modeling may be less obvious in a two-stage robust context.

We mainly have two Modeling approach: one that is arc-based [21, 50], another that is path-based [34]. Now let us examine the differences between those two kinds of modeling, regarding our elementary decisions described in section 4.5.1. In a deterministic context, both are strictly equivalent, since it is always possible to convert a path-based solution in an arc-based solution, and an arc-based solution into several path-based solutions. But actually, the path-based formulation embeds more information than the arc-based one. Indeed, in an arc-based formulation, the routing choice is let completely free, as we only manage fractions of routing paths, "non connected ducts of fibers". The main difference between these two models lies in the elementary decisions (1f), (3f), (4f) and, consequently,(3s). Indeed, in an arc-based formulation, these decisions are not *explicitly taken*, since it is supposed we can take them later anyway in a deterministic context. To

put it in another way, it is like arc-based variables contains more possible decisions than path-based decisions.

These differences being highlighted, let us examine what the impact in a 2 stage context is. Let us suppose that (PON^{2stage}_{rob}) is modelled only by the means of an arc-based formulation. In the set of first stage variables, we aim to decide where fibers should be put and what splitter rooms shall be opened. With an arc-based formulation, a network will be proposed with, as explained before, an implicit set of possible routing path for each pair of nodes that owns at least a fiber path between them. However, these implicit decisions are not really taken in the first stage of our problem. Therefore, once in the recourse problem, when the demand is revealed, it appears as these decisions are still free to make for the optimization.

As presented in section 4.5.1, first stage decisions should be (1s), (1f), (2f) and (3f). However, in an arc-based formulation, implicit decisions are (1f), (3f), (4f) and, consequently, (3s). Therefore, it means that decisions (1f) and (3f) are actually delayed in the future, once the demand is revealed. Thus, it does not seem to fit our general model (PON^{2stage}_{rob}). Yet, one could argue that it does not change the final solution, as in a deterministic context. Figure 4.3 provides a counter example that clearly exposes the difference.

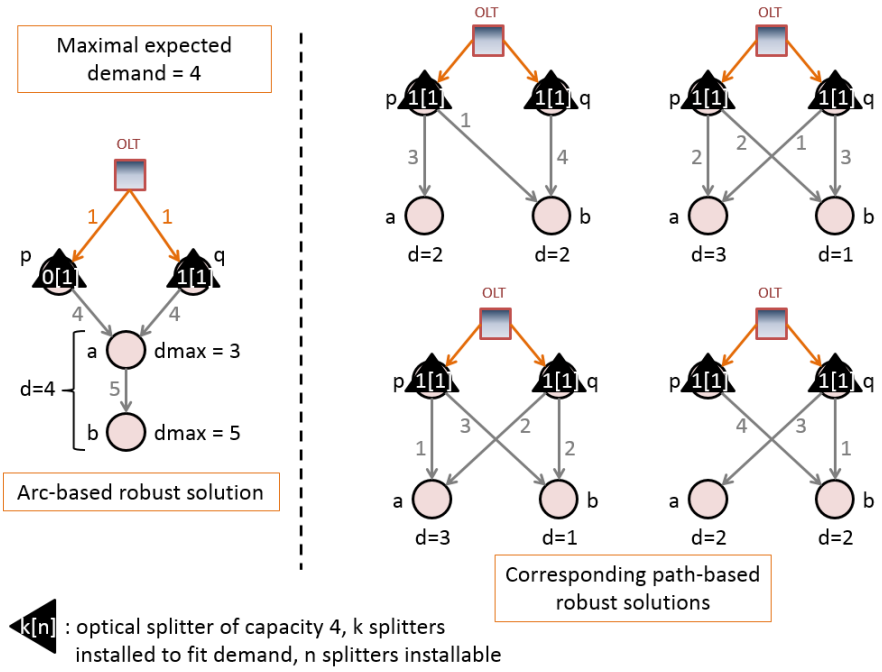

Figure 4.3: Example of a case where an arc-based solution and its corresponding path-based solution have not the same robust *splitter* cost

On this example, the splitter capacity is 4 and the total expected demand $\bar{d}$ equals 4. The left figure presents an arc based solution where only the number of fibers per duct is indicated. The maximum number of splitters one will have to install is only 1, since whatever the demand repartition is, there is always a way of branching fibers with only one splitter. On the right side, the 4 possible path-based solutions derived from the arc-based one on the left are shown. Below nodes a and b, demands that are corresponding to a so-called worst case scenario *in terms of splitters* are shown. For all the path-based solutions, in the worst case, 2 splitters will be needed to supply the worst case scenario. Therefore, the delaying of the implicit decisions in an arc-based model leads to the solving of a completely different problem. Of course, this version of the problem is of little interest in practice. Indeed, it implies that one has to deploy cut pieces of fibers along ducts without welding them together. That activity being delayed in order to supply the demand, which is too costly in terms of human intervention, but also in terms of optical budget since each weld between two fibers reduces the optical signal strength.

Modeling of the problem via a path-based formulation

Using the notations for describing the data of the problem defined in the previous sections, let us denote by $\mathbf{f}_{ij}$ the number of level 2 fibers routed along a shortest path from a node i to a node j, for any $(i,j) \in V^{*2}$, *before* the demand is revealed (i.e. so that every household is connected to the OLT). By $\mathbf{z}_i$ the maximum number of optical splitters node $i \in V^*$ can receive in the future (i.e. the number of splitters one would install if all the fibers were to be used). $\mathbf{f}_{ij}$ and $\mathbf{z}_i$ are first stage decision variables. Let $\boldsymbol{\zeta}_i$ be the number of splitters that will be actually installed on node i in order to match the revealed demand, and $\boldsymbol{\varphi}_{ij}$ the number of level 2 fiber paths that will be lighted, or used, from i to j, once the demand is revealed. $\boldsymbol{\zeta}_i$ and $\boldsymbol{\varphi}_{ij}$ are recourse variables.

We take 2 costs into account in our optimization. First, the routing cost of a fiber is denoted by c_{ij} for all $(i,j) \in V^2$. Therefore, each variable $\mathbf{f}_{ij}$ will be associated to a cost c_{ij}. But there are level 1 fibers that are also used to supply optical splitters, for an optical splitter located at node i, the routing cost of the corresponding level 1 fiber path is denoted by c_{0i}. Finally, the recourse splitter variables are associated to the splitter installation cost, denoted by C.

We denote by $P_G(d)$ the polyhedra of feasible passive optical networks that satisfy the demand vector d in the graph G. The two-stage robust passive optical network design

problem under demand uncertainty can thus be modelled as follows:

$$(PON_{rob}^{2stage}) \left\{ \min_{(\mathbf{f},\mathbf{z})\in P_G(d^{max})} \left(\sum_{(i,j)\in V^{*2}} c_{ij}\mathbf{f}_{ij} + \sum_{i\in V^*} c_{0i}\mathbf{z}_i + \max_{d\in\mathfrak{D}} \min_{\substack{(\boldsymbol{\varphi},\boldsymbol{\zeta})\in P_G(d) \\ \boldsymbol{\zeta}\leq\mathbf{z},\ \boldsymbol{\varphi}\leq\mathbf{f}}} \sum_{i\in V^*} C\boldsymbol{\zeta}_i \right) \right. \tag{4.15}$$

with $P_G(d)$ being defined as such:

$$(\mathbf{f},\mathbf{z}) \in P_G(d) \Leftrightarrow \begin{cases} \sum_{j\in V^*} \mathbf{f}_{ij} \leq m\mathbf{z}_i,\ \forall i \in V^* \\ \sum_{i\in V^*} \mathbf{f}_{ij} \geq d_j,\ \forall j \in V^* \\ \mathbf{z}_i \in \mathbb{N},\ \forall i \in V^* \\ \mathbf{f}_{ij} \in \mathbb{R}_+,\ \forall (i,j) \in V^{*2} \end{cases}$$

In this context, we denote by $Q(\mathbf{f},\mathbf{z})$ the recourse problem associated to the robust problem (PON_{rob}^{2stage}):

$$Q(\mathbf{f},\mathbf{z}) : \max_{d\in\mathfrak{D}} \min_{\substack{(\boldsymbol{\varphi},\boldsymbol{\zeta})\in P_G(d) \\ \boldsymbol{\zeta}\leq\mathbf{z},\ \boldsymbol{\varphi}\leq\mathbf{f}}} \sum_{i\in V^*} C\boldsymbol{\zeta}_i \tag{4.16}$$

We spent an extensive amount of time on the last sections to present motivated and justified modeling choices. Now, with regards to the literature, let us discuss some properties of the problem in order to identify some issues related to this problem.

Proposition 4.5.1. *The problem (PON_{rob}^{2stage}) verifies the full recourse property.*

Proof. First stage variables $\mathbf{f}_{ij}$ and $\mathbf{z}_i$ defines a network that can supply the demand vector d^{max}. If such solution exists, stating $\boldsymbol{\varphi} = \mathbf{f}$ and $\boldsymbol{\zeta} = \mathbf{z}$ also defines a feasible recourse solution that will be valid for any $d \in \mathfrak{D}$ since $d \leq d^{max}$ by definition. Therefore, there always exists a recourse solution that is valid for any scenario thus making the full recourse property verified. □

4.6 Probability bound for ensuring uncertainty set validity

In this section, we aim propose a method to design uncertainty sets when few probabilistic information is available. Indeed, we made the assumption in section 4.2 that the total amount of demand in an area can be known. However, that may not always be the case and we can imagine that sometimes, other kind of information may be available

to the decision-maker, such as the expected demand for each building in a given area (depending, for example, on various social and demographic factors). The purpose of this section is to propose a method for building an uncertainty set by deriving it from reduced probabilistic information.

As this approach is not limited to the only passive optical network problem, we decided to generalize it to any robust approach where such uncertainty set are needed. Therefore, notations and concepts invoked in this section are not related to the PON deployment problem, except for section 4.6.4 which applies it to the PON deployment problem.

We will focus on polyhedral uncertainty sets that are a generalization of the uncertainty set we use for our robust problem on fiber network deployment. Moreover, to the best of our knowledge, they do not have been investigated yet on that subject. Let us first consider an uncertain value d that can be contained in the set $\left[\check{d}, \hat{d}\right]$. Thus, without loss of generality, one can always state that d actually depends on a random variable ϕ that is always between 0 and 1:

$$d = \check{d} + \left(\hat{d} - \check{d}\right) \phi \tag{4.17}$$

For both generalization and normalization purposes, let us define the polyhedral uncertainty set we aim to study as such:

$$\Phi = \left\{ \phi \in [0,1]^{|S|} \left| \sum_{i \in S} p_i \phi_i \leq P \right. \right\}$$

with S being the set of uncertain values of the problem we are dealing with. To the uncertainty set Φ, we associate a set of p_i values that are coefficient, for all $i \in S$ and a single value P. Φ is a simple uncertainty set with only one equation that limits the overall variations of the uncertainty.

For example, the uncertainty set $\mathfrak{D}$ defined in definition 4.2.1 is a special case where $p_i = d_i^{max}$ and $P = \bar{d}$.

Now that a generic description of polyhedral uncertainty sets has been done, note that in practice, finding proper values for p_i coefficient or/and P values may prove difficult, as it is mentioned in the introduction of the section. Many cases can occur. Let us describe the two main situations one can think of:

- the decision-maker knows how his uncertain data is correlated, hence he can define himself the set Φ with p_i values that are already known. However, the P value remains hard to estimate and he wants a pertinent value for it.

- the decision-maker has no clue on how his uncertain data behave globally, therefore he only aims to define a Φ uncertainty set where he does not know p_i and P values

at all. Of course, fixing these values in order to remain coherent with what can happen is important to him.

We name this approach a "chance model" approach, referring to the famous "chance constraint" one that is present in the literature [7].

4.6.1 Theoretical bound for solution validity

Let us first state what we aim to deal with in this section.

Proposition 4.6.1. *The probability that the optimal solution of a robust problem using a pre-defined uncertainty set Φ is not feasible, denoted by $\mathcal{P}$, is bounded by*

$$\mathcal{P} \leq Prob\left(\phi \notin \Phi\right)$$

Proof. By definition, an optimal solution of a robust problem is valid for every realization of the uncertain vector ϕ within the uncertainty set Φ. Therefore, the probability that such solution is not valid in practice is at most the probability that the uncertain vector ϕ does not belong to its uncertainty set. Note that even if $\phi \notin \Phi$, it does not automatically imply that the solution will not be feasible, there is just no guarantee any more. □

Corollary 4.6.1.

$$\mathcal{P} \leq Prob\left(\sum_{i \in S} p_i \phi_i > P\right)$$

Estimating the probability that the uncertain vector belongs to its uncertainty set may seem strange, but one should keep in mind that in this section, we are dealing with cases where there is not enough relevant information to design the uncertainty set properly. Hence, in this context, there is a chance that in the end, the actual scenario can be outside the defined uncertainty set. This is what we aim to avoid by having a probability that bounds the chances that this case occurs. However, estimating this probability could prove to be impossible without a few not too restrictive assumption on random variables ϕ_i.

Hypothesis 2. *For the rest of this section, we assume that random variables ϕ_i are independent and that we know their average value, denoted by $E(\phi_i) = \theta_i < 1$ (without loss of generality).*

Please note that there are no assumption made at all on random variables distribution which may be continuous or not, and especially not always symmetric (in opposition to what is generally made in the literature, see [15] for example). Without minimal probabilistic information like the average value, it could be hard to derive anything in

order to find a useful method. To the least, the most restrictive assumption we are making here is the independence of random variables. In some problems, it could indeed be too strong to be taken lightly. However, this aspect is highly depending on the problem, and we estimated that in most cases, this would be an approximation that most decision-makers would make in practice.

Theorem 4.6.1. *Under the hypothesis 2,*

$$Prob\left(\sum_{i\in S} p_i\phi_i > P\right) \leq \min_{\lambda>0} g(\lambda) = \min_{\lambda>0}\left(\prod_{i\in S} h_i(\lambda)\right)$$

with

$$\begin{aligned} h_i(\lambda) &= e^{\lambda p_i(1-\Omega)}\left(1-\lambda p_i(1-\theta_i)e^{-\frac{\lambda p_i}{2}}\right) && \text{if } p_i > 0 \\ &= e^{-\lambda p_i\Omega}\left(1+\lambda p_i\theta_i e^{\frac{\lambda p_i}{2}}\right) && \text{if } p_i < 0 \end{aligned}$$

and

$$\Omega = \frac{P^k}{\sum_{i\in S^k} p_i}$$

Before proving the theorem, we first need to recall the following lemma, drawn from Chebytchev inequality, as it is demonstrated in the paper of Babonneau et al. in the section 8.1 of [2].

Lemma 2. *Given X a random variable:*

$$Prob(X \geq a) \leq e^{-a}E(e^X)$$

Proof.

$$\begin{aligned} E\left(e^X\right) &= E\left(e^X|X<a\right)Prob(X<a) + E\left(e^X|X\geq a\right)Prob(X\geq a) \\ &\geq E\left(e^X|X\geq a\right)Prob(X\geq a) \quad \left(\text{because } e^X>0 \Rightarrow E\left(e^X|X<a\right)\geq 0\right) \\ &\geq e^a Prob(X\geq a) \end{aligned}$$

That last inequality being from

$$E\left(e^x|X\geq a\right) \geq E\left(e^a|X\geq a\right) = e^a$$

□

Theorem 4.6.1 can thus be proved:

Proof. We denote by $S_- \subseteq S$ (respectively $S_+ \subseteq S$)the set of uncertain data associated with a negative (respectively positive) coefficient p_i in the uncertainty set Φ. Note that without loss of generality, we have $S_+ \bigcap S_- = \emptyset$ and $S_+ \bigcup S_- = S$. Let us introduce the following variable change for random variables ϕ_i by stating $\phi_i' = a_i\phi_i + b_i$ with $a_i \geq 0$ and $\phi_i' \in [-1, 1]$. Thus one can write $p_i\phi_i = \frac{p_i}{a_i}\phi_i' - \frac{p_i}{a_i}b_i$. Then, by lemma 2,

$$\begin{aligned} Prob\left(\sum_{i\in S} p_i\phi_i > P\right) &\leq Prob\left(\sum_{i\in S} \frac{p_i}{a_i}\phi_i' \geq P + \sum_{i\in S}\frac{p_i}{a_i}b_i\right) \\ &\leq e^{-\left(P+\sum_{i\in S}\frac{p_i}{a_i}b_i\right)} E\left(e^{\sum_{i\in S}\frac{p_i}{a_i}\phi_i'}\right) \\ &= e^{-\left(P+\sum_{i\in S}\frac{p_i}{a_i}b_i\right)} \prod_{i\in S} E\left(e^{\frac{p_i}{a_i}\phi_i'}\right) \end{aligned}$$

because ϕ_i variables are independent. Thus, we have for all $i \in S_+$:

$$\begin{aligned} E\left(e^{\frac{p_i}{a_i}\phi_i'}\right) &= 1 + E\left(\frac{p_i}{a_i}\phi_i'\right) + \sum_{j\geq 2} E\left(\frac{\left(\frac{p_i}{a_i}\phi_i'\right)^j}{j!}\right) \\ &\leq 1 + \frac{p_i}{a_i}\left(a_i\theta_i + b_i\right) + \sum_{j\geq 2} E\left(\frac{\left(\frac{p_i}{a_i}\right)^j}{j!}\right) \text{ because } p_i \geq 0 \\ &= 1 + \frac{p_i}{a_i}\left(a_i\theta_i + b_i\right) + e^{\frac{p_i}{a_i}} - 1 - \frac{p_i}{a_i} \\ &= \frac{p_i}{a_i}\left(a_i\theta_i + b_i - 1\right) + e^{\frac{p_i}{a_i}} \end{aligned}$$

Using the same reasoning, we get for all $i \in S_-$:

$$E\left(e^{\frac{p_i}{a_i}\phi_i'}\right) = \frac{p_i}{a_i}\left(a_i\theta_i + b_i + 1\right) + e^{\frac{-p_i}{a_i}}$$

By reintroducing in the initial expression,

$$\begin{aligned} Prob\left(\sum_{i\in S} p_i\phi_i > P\right) \leq e^{-\left(P+\sum_{i\in S}\frac{p_i}{a_i}b_i\right)} &\prod_{i\in S_+}\left(\frac{p_i}{a_i}\left(a_i\theta_i + b_i - 1\right) + e^{\frac{p_i}{a_i}}\right) \\ &\prod_{i\in S_-}\left(\frac{p_i}{a_i}\left(a_i\theta_i + b_i + 1\right) + e^{\frac{-p_i}{a_i}}\right) \end{aligned}$$

Let us remind that since $\phi_i' \in [-1, 1]$, then a_i and b_i must verify $a_i + b_i \leq 1$ and $-1 \leq b_i$. Let us state that $a_i + b_i = 1$ in the following. It thus implies that $\frac{b_i}{a_i} = -1 + \frac{1}{a_i}$, so we have $a_i\theta_i + b_i - 1 = a_i\theta_i - a_i = a_i(\theta_i - 1)$ and $a_i\theta_i + b_i + 1 = a_i(\theta_i - 1) + 2$. By replacing

in the equation, we get

$$\begin{aligned}
Prob\left(\sum_{i\in S} p_i\phi_i > P\right) &\leq e^{-\left(P+\sum_{i\in S} p_i\left(-1+\frac{1}{a_i}\right)\right)} \prod_{i\in S_+}\left(p_i(\theta_i-1)+e^{\frac{p_i}{a_i}}\right) \\
&\qquad \prod_{i\in S_-}\left(p_i\left(\theta_i-1+\frac{2}{a_i}\right)+e^{\frac{-p_i}{a_i}}\right) \\
&= e^{-P+\sum_{i\in S} p_i}\prod_{i\in S} e^{-\frac{p_i}{a_i}}\left(p_i(\theta_i-1)+e^{\frac{p_i}{a_i}}\right)\prod_{i\in S_-} e^{-\frac{p_i}{a_i}}\left(p_i\left(\theta_i-1+\frac{2}{a_i}\right)+e^{\frac{-p_i}{a_i}}\right) \\
&= e^{-P+\sum_{i\in S} p_i}\prod_{i\in S}\left(1-p_i(1-\theta_i)e^{-\frac{p_i}{a_i}}\right)\prod_{i\in S_-} e^{-\frac{2p_i}{a_i}}\left(1+p_i\left(\theta_i-1+\frac{2}{a_i}\right)e^{\frac{p_i}{a_i}}\right)
\end{aligned}$$

Since $(1-\theta_i)\geq 0$, $a_i \leq 2$ one can easily check that the value for each a_i is the one that minimizes the whole product. This value is the biggest possible for a_i, which is 2 (for $b_i=-1$). In the end, we do have

$$\begin{aligned}
Prob\left(\sum_{i\in S} p_i\phi_i > P\right) &\leq e^{-P+\sum_{i\in S} p_i}\prod_{i\in S_+}\left(1-p_i(1-\theta_i)e^{-\frac{p_i}{2}}\right) \\
&\qquad \prod_{i\in S_-} e^{-p_i}\left(1+p_i\theta_i e^{\frac{p_i}{2}}\right)
\end{aligned}$$

For any $\lambda > 0$,

$$\begin{aligned}
Prob\left(\sum_{i\in S} p_i\phi_i > P\right) &= Prob\left(\lambda\sum_{i\in S} p_i\phi_i > \lambda P\right) \\
&\leq e^{\lambda\left(-P+\sum_{i\in S} p_i\right)}\prod_{i\in S_+}\left(1-\lambda p_i(1-\theta_i)e^{-\frac{\lambda p_i}{2}}\right) \\
&\qquad \prod_{i\in S_-} e^{-\lambda p_i}\left(1+\lambda p_i\theta_i e^{\frac{\lambda p_i}{2}}\right) \\
&= \prod_{i\in S_+} e^{\lambda p_i(1-\Omega)}\left(1-\lambda p_i(1-\theta_i)e^{-\frac{\lambda p_i}{2}}\right) \\
&\qquad \prod_{i\in S_-} e^{-\lambda p_i\Omega}\left(1+\lambda p_i\theta_i e^{\frac{\lambda p_i}{2}}\right)
\end{aligned}$$

stands true with $\Omega = \dfrac{P}{\sum_{i\in S} p_i}$, especially for the λ that minimizes the right-hand side of the equation. □

Giving an analytic solution to the minimizing problem that theorem 4.6.1 yields is not easy in the general case (with $\theta_i < 1\ \forall i$). However, it is still possible to give a brief description of h_i and g functions characteristics.

Proposition 4.6.2. *h_i functions are smooth on $\mathbb{R}$, for all i.*

Proof. The proof is immediate, since every h_i function is a product of continuous functions with derivatives of all orders. □

Corollary 4.6.2. *The function g is smooth on $\mathbb{R}$.*

Proposition 4.6.3. *h_i functions are strictly positive on $\mathbb{R}$, for all i.*

Proof. The proof is given for h_i functions for which $p_i > 0$, as it also immediately applies to the other case. For negative values of λ, as for $\lambda = 0$, $h_i(\lambda)$ is obviously positive. For positive values however, one has to check if the term $\left(1 - \lambda p_i(1-\theta_i)e^{-\frac{\lambda p_i}{2}}\right)$ is always positive, noting that its value in $\lambda = 0$ is 1, as for its limit when λ tends towards infinity. The derivative of that term being:

$$\left(1 - \lambda p_i(1-\theta_i)e^{-\frac{\lambda p_i}{2}}\right)' = p_i(1-\theta_i)e^{-\frac{\lambda p_i}{2}}\left(\frac{\lambda p_i}{2} - 1\right)$$

It only nullifies for $\lambda = \dfrac{2}{p_i}$. Thus, this term do have a single extremum on $\mathbb{R}_+$, of value $1 - 2(1-\theta_i)e^{-1}$. That extremum being a minimum since the derivative takes a negative value in 0, having the term strictly positive is equivalent to having $\theta_i > 1 - \dfrac{1}{2e^{-1}}$, which is always true since, by definition, θ_i values are all positive. □

Corollary 4.6.3. *The function g is strictly positive on $\mathbb{R}$.*

In order to get an idea of what functions h_i are like, figure 4.4 provides a graphic representations of 3 of these functions, instantiated with different values of Ω and θ_i. Considering the coefficient $p_i = 1$ for all i in this case:

- $h_1(\lambda)$ is such that $\Omega = 0.3$ and $\theta_1 = 0.2$.
- $h_2(\lambda)$ is such that $\Omega = 0.7$ and $\theta_2 = 0.2$.
- $h_3(\lambda)$ is such that $\Omega = 0.7$ and $\theta_3 = 0.5$.

4.6.2 The bound in the specific case where p_i values are the same and random variables have the same average value

In some cases, there is not enough information available to give pertinent values for p_i coefficients. In that case, making them equal does not really matter. It actually makes the uncertainty set simpler. When average values are also all equal, then the writing of function g can be very simplified.

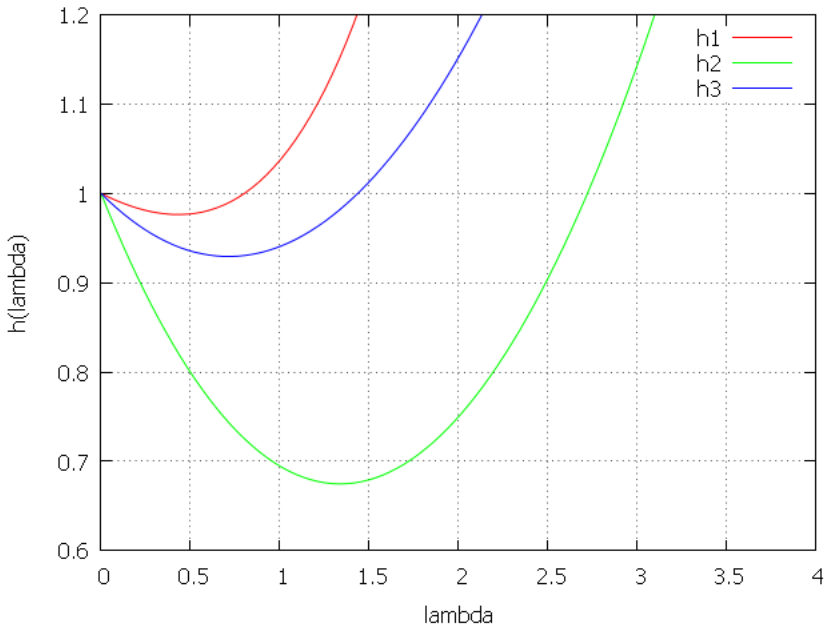

Figure 4.4: Graphic representation of functions h_1, h_2 and h_3

Hypothesis 3. *In this subsection, let us assume that p_i, and θ_i, values are all the same, so one can write:*

$$p_i = \bar{p} > 0 \text{ and } \theta_i = \bar{\theta}, \ \forall i \in S$$

Under this assumption, it is rather obvious that h_i functions are all identical, so that one can write:

$$h_i(\lambda) = h(\lambda), \ \ \forall i \in S$$

In that case, the following theorem gives an analytic solution to the minimization of g.

Theorem 4.6.2. *The minimizing problem, under the hypothesis 3, is*

$$\min_{\lambda>0} g(\lambda) = (h(\lambda))^{|S|}$$

and its optimal solution is

$$\begin{aligned}
\lambda^* &= \frac{2\log\left(2(1-\bar{\theta})\right)}{\bar{p}} && \text{for } \Omega = \frac{1}{2} \text{ and } \bar{\theta} < \frac{1}{2} \\
&= \frac{1+(1-2\Omega)W_0\left(\frac{1-\Omega}{(2\Omega-1)(1-\bar{\theta})}e^{\frac{1}{2\Omega-1}}\right)}{\left(\Omega-\frac{1}{2}\right)\bar{p}} && \text{for } \Omega > \frac{1}{2} \text{ and } \Omega > \bar{\theta} \\
&= \frac{1+(1-2\Omega)W_{-1}\left(\frac{1-\Omega}{(2\Omega-1)(1-\bar{\theta})}e^{\frac{1}{2\Omega-1}}\right)}{\left(\Omega-\frac{1}{2}\right)\bar{p}} && \text{for } \bar{\theta} < \Omega < \frac{1}{2} \text{ if } \bar{\theta} < \frac{1}{2}
\end{aligned}$$

where W is the Lambert-function, which is the inverse function of we^w.
If $\Omega \leq \bar{\theta}$, there is no optimal solution to this problem and $g(\lambda) \geq 1$, $\forall \lambda \in \mathbb{R}_+$.

Proof. A necessary condition for g to be at its minimum with $\lambda > 0$ is $g'(\lambda) = 0$ (one can easily check that g is a convex function in λ and that it possesses a single minimum for $\lambda > 0$). Since $g'(\lambda) = |S| \left(h(\lambda)\right)^{|S|-1} h'(\lambda)$ and that $h(\lambda) > 0$ $\forall \lambda > 0$ (by Proposition 4.6.3), it implies that $g'(\lambda) = 0 \Leftrightarrow h'(\lambda) = 0$. Let us denote by $\rho = \lambda \bar{p}$ so that $h(\rho) = e^{\rho(1-\Omega)} \left(1 - \rho(1 - \bar{\theta} e^{-\frac{\rho}{2}}\right)$.

Then $h'(\rho) = e^{\rho(1-\Omega)} \left(1 - \Omega + (1 - \bar{\theta}) \left(-1 + \left(\Omega - \frac{1}{2}\right) \rho\right) e^{-\frac{\rho}{2}}\right)$. Thus one has to solve the following equation

$$h'(\rho) = 0 \Leftrightarrow 1 - \Omega + (1 - \bar{\theta}) \left(-1 + \left(\Omega - \frac{1}{2}\right) \rho\right) e^{-\frac{\rho}{2}} = 0$$

If $\Omega = \frac{1}{2}$, the equation simplifies and we get $\rho = 2 \log\left(2(1 - \bar{\theta})\right)$ (which is possible since we assumed $\bar{\theta} < 1$). From $\rho = \lambda \bar{p}$ we get λ, that is positive only when $\bar{\theta} < \frac{1}{2}$.

If $\Omega \neq \frac{1}{2}$, we can divide the equation by $-2 \left(\Omega - \frac{1}{2}\right)$, thus we get

$$\frac{1 - \Omega}{-2\left(\Omega - \frac{1}{2}\right)} + (1 - \bar{\theta}) \left(\frac{1}{2\left(\Omega - \frac{1}{2}\right)} - \frac{\rho}{2}\right) e^{-\rho/2} = 0$$

Let us denote by $w = \frac{1}{2\left(\Omega - \frac{1}{2}\right)} - \frac{\rho}{2}$, thus we obtain the equation in w

$$\frac{1 - \Omega}{-2\left(\Omega - \frac{1}{2}\right)} + (1 - \bar{\theta}) w e^{w - \frac{1}{2\left(\Omega - \frac{1}{2}\right)}} = 0$$

from which we get

$$we^w = \frac{1 - \Omega}{(2\Omega - 1)(1 - \bar{\theta})} e^{\frac{1}{(2\Omega - 1)}}$$

The inverse function of we^w is known as the Lambert function, denoted by W, discovered by Lambert in 1758 and later by Euler, and first studied in [44]. By denoting $A = \frac{1 - \Omega}{(2\Omega - 1)(1 - \bar{\theta})} e^{\frac{1}{(2\Omega - 1)}}$, we deduce that $w = W(A)$, and finally, that $\rho = \frac{1}{\Omega - \frac{1}{2}} - 2W(A)$. From $\rho = \lambda \bar{p}$ we get λ.

The Lambert function W is only defined on $\left[-\frac{1}{e}, +\infty\right[$, and for $-\frac{1}{e} \leq A < 0$, $W(A)$ has two images $W_0(A)$ and $W_{-1}(A)$ (as shown on figure 4.5). In that case, there are two potential solutions λ_0^* and λ_{-1}^*. If both are positive, the one that gives the best value to g shall be chosen. If only one is positive, there is no choice. If both are negative, there is no solution, which means that g is increasing on $\mathbb{R}_+$ and as $g(0) = 1$, $g(\lambda) \geq 1$ for all

$\lambda \in \mathbb{R}_+$ in that case. If $A > 0$, then there is only one value for $W(A)$ that is given by $W_0(A)$.

The optimal solution thus highly depends on the value of Ω. When $\Omega > \frac{1}{2}$, $A > 0$ and is decreasing in Ω. The only valid branch of the Lambert function in that case is $W_0(A)$, which is increasing in $A > 0$, thus decreasing in Ω. One can easily see that λ is thus increasing with Ω. If $\Omega = \bar{\theta}$, then $W_0(A) = 0$. Therefore, Ω must be strictly greater than $\bar{\theta}$, otherwise there is no solution. The same reasoning may be applied to deduce the other cases. □

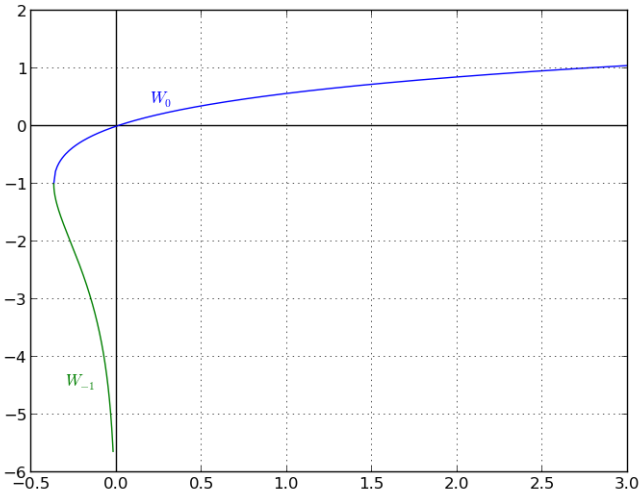

Figure 4.5: Branches of the Lambert W-function (Source: Wikipedia)

So far, this is the only case where we managed to find an analytic solution to the minimization problem in λ. However, we believe that this case can be of interest, especially when the decision-maker does not really know what values he could assign to p_i values. If it occurs that random variables all have the same average value (for example in the case of a symmetric distribution), then the decision-maker could just take p_i values that are all equal in order to quickly compute the probability, thus fixing an acceptable P value.

4.6.3 A routine to compute an approximation of the bound in the general case

As in the general case, we were not able to provide an analytic solution to the problem $\min_{\lambda>0} g(\lambda)$, we aim to propose an algorithm that could propose a numerical approximation of the bound with a high enough precision.

Proposition 4.6.4. *The function g has a minimum if, and only if,*

$$\sum_{i \in S} p_i \theta_i < P$$

Otherwise, it is an increasing function that takes values strictly superior to 1 for $\lambda > 0$.

Proof. The derivative of function g is:

$$g'(\lambda) = \sum_{i \in S} h_i'(\lambda) \prod_{j \in S \setminus \{i\}} h_j(\lambda)$$

As given by Proposition 4.6.3, h functions are strictly positive on $\mathbb{R}_+$. However, they do not always are increasing, depending on Ω and θ_i values, as shown by theorem 4.6.2. One can easily see that g tends towards $+\infty$ as λ increases. As each h' function is convex in λ over $\mathbb{R}$, the condition for g having a minimum on $\mathbb{R}_+$ is that the derivative of g in 0 is negative. Otherwise, g would be an increasing function.

$$\begin{aligned} g'(0) &= \sum_{i \in S} h_i'(0) \prod_{j \in S \setminus \{i\}} h_j(0) \\ &= \sum_{i \in S} p_i(1 - \Omega) - p_i(1 - \theta_i) \end{aligned}$$

Therefore, having $g'(0) < 0$ imposes that $\sum_{i \in S} p_i(\theta_i - \Omega) < 0$, that is:

$$\begin{aligned} \sum_{i \in S} p_i \theta_i &< \Omega \sum_{i \in S} p_i \\ &< P, \quad \text{because } \Omega = \frac{P}{\sum_{i \in S} p_i} \end{aligned}$$

□

Remark 4.6.1. *The latter proposition shows that the value chosen for P should always be greater than the expected value of $\sum_{i \in S} p_i \phi_i$, otherwise the probability bound will always be greater than 1.*

The minimization problem yielded in theorem 4.6.1 consists in minimizing a function of a single variable. Under the assumptions and properties demonstrated above, one can easily check that the function g possesses a minimum in $\lambda > 0$ if the condition given by Proposition 4.6.4 is met. Finding this minimum consists in finding the zero of its derivative, which can be done by classic numerical methods, such as the famous Newton-Raphson's method (described in the bibliographical section 1.3.1). Algorithm 5 gives a numerical routine for finding a good approximation of our bound. Note that the uniqueness of the minimum is not proven in our case, because the product of convex

positive functions is not immediately convex. However, in most cases, we conjecture that there will be a unique minimum point. If not, then another starting point could be used for the method as algorithm 5 as it is defined only finds the "first" minimum it can reach from $\lambda = 0$.

As the derivative in $\lambda = 0$ is considered negative, it is the starting point of the method. Then, the next points are computed iteratively, until the gap between the current value of the derivative and 0 is less than a defined ϵ. This algorithm offers nice convergence properties, and the next λ value may be computed in $o(|S|^3)$. Numerical errors can occur and it may require a little tuning.

Algorithm 5 Computation of an approximation of the probability bound in the general case

Require: Ω, $\theta_i, \forall i \in S$, ϵ
Require: λ, λ_{next}, X_{slack}
1: $\lambda \leftarrow 0$
2: $X_{slack} \leftarrow +\infty$
3: **while** $X_{slack} > \epsilon$ **do**
4: $\quad \lambda_{next} = \lambda - \dfrac{\sum_{i \in S} h_i'(\lambda) \prod_{j \in S \setminus \{i\}} h_j(\lambda)}{\sum_{i \in S} h_i''(\lambda) \prod_{j \in S \setminus \{i\}} h_j(\lambda) + h_i'(\lambda) \sum_{j \in S \setminus \{i\}} h_j'(\lambda) \prod_{k \in S \setminus \{i,j\}} h_k(\lambda)}$
5: $\quad X_{slack} \leftarrow |g'(\lambda_{next})|$
6: $\quad \lambda \leftarrow \lambda_{next}$
7: **end while**
8: **return** λ

4.6.4 Bound computing and estimation of the bound's quality

For the sake of testing purposes, let us suppose we are dealing with a problem where the uncertainty set is modeled as such:

$$\Phi_{test} = \left\{ \phi \in [0,1]^{|S|} \,\middle|\, \sum_{i \in S} 100\phi_i \leq P \right\}$$

We aim to define P so that the probability that $\sum_{i \in S} 100\phi_i$ exceeds it is very low. We suppose that random variables ϕ_i are all independent and that their average value is known, and is always 0.2 for all $i \in S$. Therefore, Theorem 4.6.2 can be used to compute the bound for $\Omega = \dfrac{P}{100}$ going from 0.2 to 1. The result of such computation are shown in figure 4.6, for $|S| \in \{20, 50, 100\}$.

As it is shown on the figure, the probability bound decreases relatively fast as Ω (therefore P) increases. This was to be expected since the chances of being out of smaller

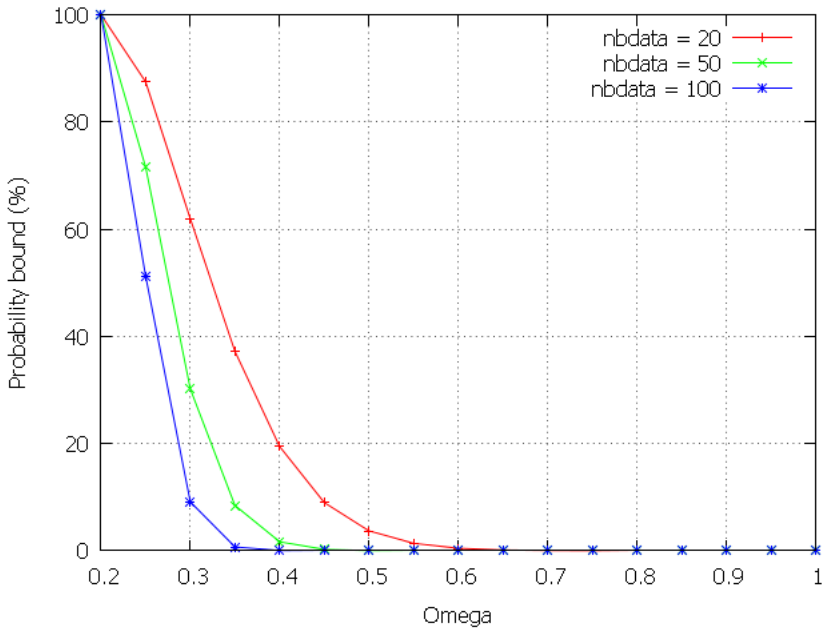

Figure 4.6: Probability bound in function of Ω for 3 different sizes of uncertainty sets

uncertainty set should be higher. Moreover, the decrease of the bound gets sharper as the number of uncertain data increases. This is due to the fact that as $|S|$ increases, there are less chances for uncertain data to increase all at the same time. This is on that specific point that the random variables independence is important.

In order to estimate the precision of algorithm 5, we applied to the cases tested above. Empirically, there are 3 parameters that tend to increase the number of step necessary for the algorithm to stop.

- The ϵ parameter: as it goes smaller, more steps are needed.
- The number of uncertain data $|S|$: as it goes bigger, more steps are needed.
- The value of P: as it goes bigger, more steps are needed.

Table 4.1 shows the performance of algorithm 5, based on the same uncertainty set Φ_{test} presented above. The algorithm is applied for the same 3 values of $|S|$ and for 4 values of ϵ. In addition, the algorithm runs for 3 different values of $\Omega \in \{0.3, 0.5, 0.7\}$. For each case, we provide the following information (in that order in the table):

- The absolute gap between the probability bound found by the algorithm and the exact value found using Theorem 4.6.2. It is set to 0% if it is strictly less than 0.01%.

		$\|S\|$		
		20	50	100
ϵ	10	0.1% - 2 steps - 13 ms 0.15% - 5 steps - 22 ms 0.99% - 5 steps - 24 ms	0% - 4 steps - 175 ms 0.4% - 6 steps - 239 ms 0.3% - 6 steps - 252 ms	0.03% - 5 steps - 1.5 s 0.3% - 6 steps - 2 s 0.1% - 7 steps - 2.1 s
	1	0% - 4 steps - 19 ms 0% - 6 steps - 29 ms 1.04% - 6 steps - 29 ms	0% - 5 steps - 207 ms 0.06% - 8 steps - 328 ms 0.06% - 8 steps - 287 ms	0% - 6 steps - 1.8 s 0.02% - 9 steps - 2.7 s 0.01% - 9 steps - 3 s
	0.1	0% - 5 steps - 21 ms 0% - 7 steps - 38 ms 0% - 10 steps - 42 ms	0% - 6 steps - 258 ms 0.01% - 10 steps - 422 ms 0% - 11 steps - 457 ms	0% - 7 steps - 2.1 s 0% - 11 steps - 3.3 s 0% - 11 steps - 3.6 s
	0.01	0% - 6 steps - 28 ms 0% - 7 steps - 33 ms 0% - 12 steps - 47 ms	0% - 7 steps - 331 ms 0% - 12 steps - 550 ms 0% - 13 steps - 544 ms	0% - 9 steps - 2.8 s 0% - 13 steps - 3.9 s 0% - 14 steps - 4.2 s

Table 4.1: Table of performance and comparison of the numerical approximation algorithm

- The number of iterations necessary to the algorithm to finish.
- The time needed for the algorithm to finish.

The main observation is that the absolute gap rarely exceeds 1%, even for larger values of ϵ. Regarding the small number of steps needed to finish, we can estimate that the convergence speed is high. Of course, the algorithm is dealing here with a function g that may be very different from the one in the general case (that is, without assumptions of hypothesis 3). However, as it is the only case where we can have the exact value of the probability bound, it is our only comparison basis.

Conclusion

In conclusion of this chapter, we examined various modeling options, by building, step by step, a model that is coherent with field expectations. After defining a pertinent uncertainty set for our problem, we first proposed a single stage robust approach for the passive optical network design problem. We explored different Modeling for this problem, thus highlighting their respective assets and limitations.

Robust single stage optimization's potential greatly depends on the mathematical formalism used to define the studied problem. In our case, after performing some tests, we estimated that those limitations were too important for these models to be satisfying. Therefore, we moved on to two-stage robust optimization to tackle our problem.

We discussed in detail the methods behind two-stage optimization, recalling that while creating a robust mathematical model from a deterministic one, one should always start to build it from actual *decisions* instead of *variables*, which can embed (as this is the case in our context, some hidden or implicit information. We applied this reasoning to compare arc-based and path-based approaches and we concluded that path-based formulations were more suited for our Modeling intent.

Moreover, we proposed a method for modeling the uncertainty set in the case where probabilistic information is available by providing probability bounds on the set's validity.

In the end we defined a two stage robust problem, (PON_{rob}^{2stage}). Our next goal is to solve this problem.

Chapter 5

Development of exact solving approaches for the two-stage robust passive optical network design problem under demand uncertainty

Introduction

In chapter 4, a two-stage robust model for solving the passive optical network design problem under demand uncertainty has been proposed. This problem, denoted by (PON_{rob}^{2stage}), presents various difficulties. The first one being that this is a two-stage problem, under the form of a "min-max-min" formulation. By itself, it usually leads to very large linear formulations, thus making its solving a serious issue even when all variables are continuous.

In addition to that, having integer recourse variables makes the problem much harder, and not only because of variable integrity, but mainly because there is almost nothing that is proposed in the literature to solve these problems to optimality. Actually, the only reference one could find, to the best of our knowledge, is a work of Zhao and Zeng [55] published in 2012 on Optimization Online. Hence, as the problem can legitimately be expected to be hard to solve, the way of doing it also represents a challenge.

Regarding the difficulties that await, one could feel that the efforts needed to design such method will probably lead to a solving approach that will have a limited spectrum of efficiency (in terms of problem size that can be solved). However, these results should provide a lot of insight on what makes a solution robust, thus allowing us to later design approached methods based on the knowledge we may gather from this study. We thus considered that having such method would be worth the effort.

The objective of this chapter is to design one or several exact solving methods to find the optimal solution of problem (PON_{rob}^{2stage}) and, from these solutions, to gain insight on what makes a solution robust.

5.1 A general column-and-constraint algorithm for solving the master problem

Before giving a method to solve the robust problem we propose to rewrite it. Similarly to what is presented in the bibliographical study on two-stage robust problems of section 1.2.6, to each possible value of $\boldsymbol{d}$ in the uncertainty set $\mathfrak{D}$ corresponds a scenario and we have: $\mathfrak{D} = \{\boldsymbol{d} \in \mathbb{N}^{|V|} |\ d_i \leq d_i^{max}, \forall i \in V \ \texttt{and} \ \sum_{i \in V} d_i \leq \bar{d}\} = \{\boldsymbol{d}^s, s = 1, ..., S\}$, where $\boldsymbol{d}^s$ denotes a scenario. Since $\boldsymbol{d}$, $\bar{d}$ and d_i^{max} are integer, there is a finite (but probably huge) number S of scenarios.

To each d^s we associate a splitter recourse variable $\boldsymbol{\zeta}^s$ and a fiber recourse variable $\boldsymbol{\varphi}^s$: we create S copies of the variables $\boldsymbol{\zeta}$ and $\boldsymbol{\varphi}$.

By enumerating all scenarios of demands, the problem PON_{rob}^{2stage}, that we will now denote by PON_{rob} for the sake of simplicity, can be rewritten as the following mixed integer linear program where, for shortening, we denote $\sum_{i \in V} c_i \zeta_i$ by $c\boldsymbol{\zeta}$.

$$(PON_{rob}) \begin{cases} \min\limits_{\gamma, (\mathbf{f},\mathbf{z}), (\boldsymbol{\zeta}^s, \boldsymbol{\varphi}^s)_{s=1,...,S}} \sum\limits_{(i,j) \in V^2} c_{ij}\mathbf{f}_{ij} + \sum\limits_{i \in V} c_{0i}\mathbf{z}_i + \boldsymbol{\gamma} \\ \text{s.t. } (\mathbf{f}, \mathbf{z}) \in P_G(\boldsymbol{d}^{max}) \\ \left.\begin{array}{l} \boldsymbol{\gamma} \geq\ c\boldsymbol{\zeta}^s \\ (\boldsymbol{\varphi}^s, \boldsymbol{\zeta}^s) \in P_G(\boldsymbol{d}^s) \\ \boldsymbol{\zeta}^s \leq \mathbf{z}, \boldsymbol{\varphi}^s \leq \mathbf{f} \end{array}\right\} \forall s = 1, ..., S \end{cases}$$

Let $\{\boldsymbol{\gamma}^*, \mathbf{f}^*, \mathbf{z}^*;\ \boldsymbol{\zeta}^{*s}, \boldsymbol{\varphi}^{*s}, s = 1, ..., S\}$ be an optimal solution of PON_{rob}. Then, there is a scenario $\boldsymbol{d}^{\bar{s}}$ such that $\boldsymbol{\gamma}^* = c\boldsymbol{\zeta}^{*\bar{s}}$: $\boldsymbol{d}^{\bar{s}}$ is the worst scenario for the decision variables $\mathbf{f}^*, \mathbf{z}^*$, $\boldsymbol{\varphi}^{*\bar{s}}$ and $\boldsymbol{\zeta}^{*\bar{s}}$ which are the best responses to this scenario. But S may be huge, then PON_{rob} has an exponential number of constraints and is generally intractable. For now, let us assume that we can solve $Q(\mathbf{f}, \mathbf{z})$ for any values of $\mathbf{f}$ and $\mathbf{z}$. Since considering all scenarios at once is inconceivable in practice, let us consider only a small subset $\mathfrak{D}_0$ of $\mathfrak{D}$ and then an increasing sequence (in cardinality) of subsets $\mathfrak{D}_l$ of $\mathfrak{D}$, for $l = 1, ..., L$, such that solving PON_{rob} for $\mathfrak{D} = \mathfrak{D}_l$ is tractable. For all $\mathfrak{D}_l \subseteq \mathfrak{D}$, we denote by PON_{rob}^l the restricted problem obtained by using $\mathfrak{D}_l$ as the uncertainty set instead of $\mathfrak{D}$ in the above program PON_{rob}, and by $\{\mathbf{f}^l, \mathbf{z}^l, \boldsymbol{\gamma}^l\}$ the optimal solution of PON_{rob}^l.

Let $l = 0$, and consider a given subset $\mathfrak{D}_0$. Solving PON_{rob}^0 gives a feasible solution $\{\mathbf{f}^0, \mathbf{z}^0, \gamma^0\}$ and, since only a subset of scenarios is taken into account, a lower bound LB

to PON_{rob}:

$$LB = v(PON^0_{rob}) = \sum_{(i,j)\in V^2} c_{ij}\mathbf{f}^0_{ij} + \sum_{i\in V} c_i\mathbf{z}^0_i + \boldsymbol{\gamma}^0$$

Then, by solving $Q(\mathbf{f}^0, \mathbf{z}^0)$, we get the worst scenario $\boldsymbol{d}^{s0}$ for $(\mathbf{f}^0, \mathbf{z}^0)$ and we obtain an upper bound UB of PON_{rob}:

$$UB = \sum_{(i,j)\in V^2} c_{ij}\mathbf{f}^0_{ij} + \sum_{i\in V} c_i\mathbf{z}^0_i + v(Q(\mathbf{f}^0, \mathbf{z}^0)) = LB - \boldsymbol{\gamma}^0 + v(Q(\mathbf{f}^0, \mathbf{z}^0))$$

Then if we solve the robust problem restricted to $\mathfrak{D}_1 = \mathfrak{D}_0 \cup \{\boldsymbol{d}^{s0}\}$, we (possibly) increase the value of LB and then (possibly) decrease the value of UB, and we iterate the process until $UB = LB$, as described in the column-and-constraints generation algorithm 6.

Algorithm 6 A column-and-constraint generation algorithm for solving PON_{rob} to optimality

Require: $UB = +\infty$, $LB = -\infty$, $\varepsilon \in \mathbb{R}_+$, $\mathfrak{D}_0$, $l = 0$;

1: **while** $UB - LB > \varepsilon$ **do**
2: solve PON^l_{rob}; let $\{\mathbf{f}^l, \mathbf{z}^l, \boldsymbol{\gamma}^l\}$ be an optimal solution;
3: $LB \leftarrow v(PON^l_{rob})$;
4: solve $\mathcal{Q}(\mathbf{f}^l, \mathbf{z}^l)$; let $\boldsymbol{d}^{sl}$ be an optimal solution;
5: $UB \leftarrow \min\{UB, LB - \boldsymbol{\gamma}^l + v(Q(\mathbf{f}^l, \mathbf{z}^l)\}$;
6: $\mathfrak{D}_{l+1} \leftarrow \mathfrak{D}_l + \{\boldsymbol{d}^{sl}\}$; $l \leftarrow l + 1$;
7: **end while**

At step l, in the calculation of both bounds UB and LB the fibers network is given by $\mathbf{f}^l$ and the possible splitter location by $\mathbf{z}^l$; in the solution corresponding to UB the splitter location is the best one obtained for the worse scenario of demand associated to $(\mathbf{f}^l, \mathbf{z}^l)$ while in the solution corresponding to LB the splitter location is the best one for the worse scenario among the scenarios taken into account at step l. There is a finite number of scenarios and $LB = UB$ if all the scenarios have been added: therefore the algorithm converges to an optimal solution with a finite number of iterations (at most S). Of course, we aim to add only a small subset of scenarios in order to solve the problem. This framework is not new regarding the literature. The difficulty here is to solve the recourse problem in a mixed integer variables context.

5.2 Solving the recourse problem

5.2.1 The design of a column-and-constraint generation algorithm for solving the recourse problem

We are now interested in solving $Q(\tilde{\mathbf{f}}, \tilde{\mathbf{z}})$, $\tilde{\mathbf{f}}$ and $\tilde{\mathbf{z}}$ being fixed values. This problem is modeled as a so called max-min problem with mixed integer variables. The main idea we can use is derived from the paper of Zhao and Zeng [55]: that is considering the integer variables vector to be in a given set, and apply a column-and-constraint generation algorithm to generate these vectors and their associated continuous variables in a similar way as for the master problem.

First, let us denote by Z the set of integer vectors such that:

$$Z = \left\{ \boldsymbol{\zeta}^r \,\middle|\, \boldsymbol{\zeta}^r \in \mathbb{N}^{|V|},\, \boldsymbol{\zeta}^r \leq \tilde{\mathbf{z}} \right\}$$

Therefore, we can rewrite the problem $Q(\tilde{\mathbf{f}}, \tilde{\mathbf{z}})$, first presented as in (4.16), as follows:

$$Q(\tilde{\mathbf{f}}, \tilde{\mathbf{z}}) \,:\, \max_{d \in \mathfrak{D}} \left(\min_{\boldsymbol{\zeta}^r \in Z \text{ s.t. } \exists \boldsymbol{\varphi} \in P_G(\boldsymbol{d}, \boldsymbol{\zeta}^r) \text{ and } \boldsymbol{\varphi} \leq \tilde{\mathbf{f}}} c\boldsymbol{\zeta}^r \right)$$

The minimization problem here actually consists in picking only one integer vector $\boldsymbol{\zeta}^r$ within the set Z. But that vector $\boldsymbol{\zeta}^r$ can be chosen only if the answer to the following decision problem is *yes*: "is there a $\boldsymbol{\varphi}$ such that $\boldsymbol{\varphi} \in P_G(\boldsymbol{d}, \boldsymbol{\zeta}^r)$ and $\boldsymbol{\varphi} \leq \tilde{\mathbf{f}}$?". In other words, we can only pick a splitter repartition that can supply the demand vector d and, amongst all the possible splitter repartitions, we take the cheapest one.

There is an exponential number of variables $\boldsymbol{\zeta}^r$ in Z and thus an exponential number of constraints. In order to verify if there exists $\boldsymbol{\varphi} \in P_G(\tilde{\boldsymbol{d}}, \boldsymbol{\zeta}^{\tilde{r}})$ with $\boldsymbol{\varphi} \leq \tilde{\boldsymbol{f}}$ for given values $\tilde{\boldsymbol{d}}$ and $\boldsymbol{\zeta}^{\tilde{r}}$, of $\boldsymbol{d}$ and $\boldsymbol{\zeta}^r$, we transform the decision problem "is there $\boldsymbol{\varphi}$...?" in the following optimization program:

$$(\hat{P}_{mult})(\tilde{\boldsymbol{d}}, \boldsymbol{\zeta}^{\tilde{r}}) \begin{cases} \min\limits_{(\boldsymbol{\varphi}, a^{\tilde{r}})} \sum\limits_{j \in V} a_j^{\tilde{r}} & \\ \text{s.t. } (\boldsymbol{\varphi}, a^{\tilde{r}}) \in \hat{P}_G(\tilde{\boldsymbol{d}}, \boldsymbol{\zeta}^{\tilde{r}}) \Leftrightarrow & \left| \begin{array}{l} \sum\limits_{j \in V} \boldsymbol{\varphi}_{ij} \leq m\boldsymbol{\zeta}_i^{\tilde{r}}, \forall i \in V^* \\ \sum\limits_{i \in V} \boldsymbol{\varphi}_{ij} + a_j^{\tilde{r}} \geq \tilde{d}_j, \forall j \in V^* \\ \boldsymbol{\varphi}_{ij} \leq \mathbf{f}_{ij}, \forall (i,j) \in V^{*2} \\ a^{\tilde{r}} \geq 0, \boldsymbol{\varphi}_{ij} \in \mathbb{R}^+, \forall (i,j) \in V^{*2} \end{array} \right. \end{cases}$$

There exists $\boldsymbol{\varphi} \in P_G(\tilde{\boldsymbol{d}}, \boldsymbol{\zeta}^{\tilde{r}})$ with $\boldsymbol{\varphi} \leq \mathbf{f}$ if and only if $(\hat{P}_{mult})(\tilde{\boldsymbol{d}}, \boldsymbol{\zeta}^{\tilde{r}})$ has an optimum value equal to 0, i.e. $a_j^{\tilde{r}*} = 0$, $\forall j \in V$ or $\boldsymbol{a}^{\tilde{\boldsymbol{r}}*} = 0$.

Remark 5.2.1. *There is an alternative formulation to* $(\hat{P}_{mult})(\tilde{\boldsymbol{d}}, \boldsymbol{\zeta}^{\tilde{r}})$ *which, instead of using multiple variables* $a_j^{\tilde{r}}$*, uses only one "gap" variable denoted by* $s^{\tilde{r}}$*. The new formulation is thus:*

$$(\hat{P}_{single})(\tilde{\boldsymbol{d}}, \boldsymbol{\zeta}^{\tilde{r}}) \begin{cases} \min\limits_{(\boldsymbol{\varphi}, s^{\tilde{r}})} s^{\tilde{r}} \\ s.t.\ (\boldsymbol{\varphi}, s^{\tilde{r}}) \in \hat{P}_G(\tilde{\boldsymbol{d}}, \boldsymbol{\zeta}^{\tilde{r}}) \Leftrightarrow \end{cases} \left| \begin{array}{l} \sum\limits_{j \in V} \boldsymbol{\varphi}_{ij} \leq m\boldsymbol{\zeta}_i^{\tilde{r}}, \forall i \in V \\ \sum\limits_{i \in V} \boldsymbol{\varphi}_{ij} + s^{\tilde{r}} \geq \tilde{d}_j, \forall j \in V \\ \boldsymbol{\varphi}_{ij} \leq \mathbf{f}_{ij}, \forall (i,j) \in V^2 \\ s^{\tilde{r}} \geq 0, \boldsymbol{\varphi}_{ij} \in \mathbb{R}^+, \forall (i,j) \in V^2 \end{array} \right.$$

This alternative formulation will have an impact, as shown in section 5.2.2.

In the following, we will use $(\hat{P}_{mult})$ alone but one has to remember that the following results are also valid for $(\hat{P}_{single})$.

Now, let (P_1) be the following program:

$$(P_1) \begin{cases} \max\limits_{\boldsymbol{d}} & \boldsymbol{\theta} \\ \text{s.t.} & \boldsymbol{d} \in \mathfrak{D} \\ & \boldsymbol{\theta} \leq c\boldsymbol{\zeta}^r,\ \forall \boldsymbol{\zeta}^r \in Z \text{ such that } v\left((\hat{P}_{mult})(\boldsymbol{d}, \boldsymbol{\zeta}^{\tilde{r}})\right) = 0 \end{cases} \tag{5.1}$$

Proposition 5.2.1. *The program* (P_1) *verifies*

$$v(P_1) = v\left(Q(\tilde{\mathbf{f}}, \tilde{\mathbf{z}})\right)$$

Proof. As constraint (5.1) ensures that all valid vectors $\boldsymbol{\zeta}^r$ are chosen, $\boldsymbol{\theta}$ always equals the one that minimizes $c\boldsymbol{\zeta}^r$ for a given d. □

Now, consider the following program, given for a subset Z_2 of Z:

$$(P_2) \begin{cases} \max\limits_{\boldsymbol{d}} & \boldsymbol{\theta} \\ \text{s.t.} & \boldsymbol{d} \in \mathfrak{D} \\ & \boldsymbol{\theta} \leq \min\limits_{\boldsymbol{\varphi} \text{ s.t.} (\boldsymbol{\varphi}, a^r) \in \hat{P}_G(\boldsymbol{d}, \boldsymbol{\zeta}^r)} \sum\limits_{j \in V} a_j^{\tilde{r}},\ \forall \boldsymbol{\zeta}^r \in Z_2 (\subseteq Z) \end{cases} \tag{5.2}$$

Proposition 5.2.2. *If* $Z_2 = Z$*, then we have*

$$v(P_2) = \boldsymbol{\theta}^* = 0$$

Proof. Since the robust problem verifies the full recourse property (Proposition 4.5.1), it is straightforward. □

Let us assume for now that we can solve (P_2) when constraints (5.2) are restricted to a given subset of constraints (i.e. for all $\boldsymbol{\zeta}^r \in Z_2 \subset Z$), as we will deal with this aspect later in section 5.2.2. Thus, it would be possible to select, iteratively, $\boldsymbol{\zeta}^1$, $\boldsymbol{\zeta}^2$, ..., $\boldsymbol{\zeta}^K$ in Z and solve a sequence of sub-programs (P_2^1),..., (P_2^K) of (P_2) obtained by adding successively the constraints associated to $\boldsymbol{\zeta}^1$, $\boldsymbol{\zeta}^2$, ..., $\boldsymbol{\zeta}^K$. We would only stop for the first $\boldsymbol{\zeta}^r$ such that $\boldsymbol{\theta} = 0$, i.e. $\boldsymbol{\theta}^1 > 0$,...,$\boldsymbol{\theta}^{K-1} > 0$ and $\boldsymbol{\theta}^K = 0$.

Remark 5.2.2. *We have $v(P_2) = 0$ if and only if, for any value of $\boldsymbol{d}$, there is at least one $\boldsymbol{\zeta}^{\bar{r}}$ in Z_2 such that $\boldsymbol{a}^{\bar{r}} = 0$, i.e. such that there exists $\boldsymbol{\varphi} \in P_G(\boldsymbol{d}, \boldsymbol{\zeta}^{\bar{r}})$, $\boldsymbol{\varphi} \leq \tilde{\boldsymbol{f}}$.*

Let us denote by $Z^k \subset Z$ the subset of constraints associated to $\boldsymbol{\zeta}^1$, $\boldsymbol{\zeta}^2$, ..., $\boldsymbol{\zeta}^k$ ($Z^k = \left\{\boldsymbol{\zeta}^1, ..., \boldsymbol{\zeta}^k\right\}$) and let $\boldsymbol{d}_k^*$ be the value of $\boldsymbol{d}$ obtained when solving (P_2^k), then $\boldsymbol{d}_k^*$ is the worst scenario associated to Z^k. To obtain $\boldsymbol{\zeta}^{k+1}$ we solve the following program where $\boldsymbol{d} = \boldsymbol{d}_k^*$:

$$\boldsymbol{\zeta}^{k+1} = \arg\min_{\boldsymbol{\zeta}^r} c\boldsymbol{\zeta}^r (\text{for } \boldsymbol{\zeta}^r \text{ such that } (\boldsymbol{\varphi}, \boldsymbol{\zeta}^r) \in P_G(\boldsymbol{d}_k^*), \boldsymbol{\zeta}^r \leq \tilde{\boldsymbol{z}}, \boldsymbol{\varphi} \leq \tilde{\mathbf{f}}) \tag{5.3}$$

We denote by $R(\tilde{\mathbf{f}}, \tilde{\mathbf{z}}, \boldsymbol{d}_k^*)$ this minimization problem.

Since $c\boldsymbol{\zeta}^{k+1}$ is minimal (from (5.3)) and since the problem verifies the complete recourse property, we know that $c\boldsymbol{\zeta}^{k+1}$ is finite and thus $v\left((\hat{P}_{mult})(\boldsymbol{d}_k^*, \boldsymbol{\zeta}^{k+1})\right) = 0$. Finally, we have the following theorem:

Theorem 5.2.1. *The optimal value of the recourse problem is:*

$$v(Q(\tilde{\mathbf{z}}, \tilde{\mathbf{f}})) = \max_{k \in \{1,...,K\}} c\boldsymbol{\zeta}^k$$

Proof. Let (P_1^{restr}) be the following restricted version of (P_1):

$$(P_1^{restr}) \begin{cases} \max\limits_{\boldsymbol{d}} & \boldsymbol{\theta} \\ \text{s.t.} & \boldsymbol{d} \in \mathfrak{D} \\ & \boldsymbol{\theta} \leq c\boldsymbol{\zeta}^r, \forall \boldsymbol{\zeta}^r \in \left\{\boldsymbol{\zeta}^1, ..., \boldsymbol{\zeta}^K\right\} \text{ such that } v\left((\hat{P}_{mult})(\boldsymbol{d}, \boldsymbol{\zeta}^{\tilde{r}})\right) = 0 \end{cases} \tag{5.4}$$

We are going to prove that $\max(c\boldsymbol{\zeta}^1, ..., c\boldsymbol{\zeta}^K) \leq v(Q(\tilde{\mathbf{z}}, \tilde{\mathbf{f}})) \leq v(P_1^{restr})$ and then that $\max(c\boldsymbol{\zeta}^1, ..., c\boldsymbol{\zeta}^K) = v(P_1^{restr})$.

First, from (5.3), we know that for each $k \in \{1, ..., K\}$ we have $c\boldsymbol{\zeta}^k \leq v(Q(\tilde{\mathbf{z}}, \tilde{\mathbf{f}}))$ since to obtain $\boldsymbol{\zeta}^k$, we consider only the demand $\boldsymbol{d}_k^*$ of $\mathfrak{D}$. Then $\max(c\boldsymbol{\zeta}^1, ..., c\boldsymbol{\zeta}^K) \leq v(Q(\tilde{\mathbf{z}}, \tilde{\mathbf{f}}))$. Second, since (P_1^{restr}) takes into account only a subset of constraints of (P_1), we have $v(Q(\tilde{\boldsymbol{z}}, \tilde{\boldsymbol{f}})) = v(P1) \leq v(P_1^{restr})$. We know that $v(P_1^{restr})$ is finite, i.e. if we replace Z by $\left\{\boldsymbol{\zeta}^1, \boldsymbol{\zeta}^2, ..., \boldsymbol{\zeta}^K\right\}$ in $Q(\tilde{\mathbf{z}}, \tilde{\mathbf{f}})$, we obtain a solution with a finite value. Finally, since in (P_1^{restr}) we have $\theta = c\boldsymbol{\zeta}^r$ for some $\boldsymbol{\zeta}^r \in \left\{\boldsymbol{\zeta}^1, ..., \boldsymbol{\zeta}^K\right\}$, we have $v(P_1^{restr}) \leq \max(c\boldsymbol{\zeta}^1, ..., c\boldsymbol{\zeta}^k)$, which concludes the proof. □

In other words, Theorem 5.2.1 states that by finding a subset of vectors $\boldsymbol{\zeta}^r$ such that any demand configuration can be supplied for fixed values of $\tilde{\mathbf{z}}$ and $\tilde{\mathbf{f}}$, then an optimal solution of (P_1) is amongst them. In the end, instead of solving (P_1), we just need to solve (P_2) iteratively until $v(P_2) = 0$. Assuming we can effectively solve (P_2), let us summarize the column-and-constraint generation algorithm we detailed in this section:

Algorithm 7 A column-and-constraint generation algorithm for solving $Q(\tilde{\mathbf{z}}, \tilde{\mathbf{f}})$ to optimality

Require: $UB = +\infty$, $\varepsilon \in \mathbb{R}_+$, $k = 0$;

1: find an initial vector $\boldsymbol{\zeta}^1$ by solving (5.3) for a randomly chosen scenario $\boldsymbol{d} \in \mathfrak{D}$
2: $Z_2 \leftarrow \boldsymbol{\zeta}^1$
3: **while** $UB > \varepsilon$ **do**
4: solve (P_2) with Z_2; let $\boldsymbol{d}_k$ be an optimal solution;
5: $UB \leftarrow v(P_2)$;
6: solve (5.3); let $\boldsymbol{\zeta}^{k+1}$ be an optimal solution;
7: $Z_2 \leftarrow Z_2 \cup \{\boldsymbol{\zeta}^{k+1}\}$; $k \leftarrow k + 1$;
8: **end while**
9: return $\max_{k' \in \{1,\dots,k\}} c\boldsymbol{\zeta}^{k'}$ and the associated $\boldsymbol{d}_{k^*}$ with $k^* = \arg \max_{k' \in \{1,\dots,k\}} c\boldsymbol{\zeta}^{k'}$;

5.2.2 Solving the program (P_2)

In each constraint (5.2) of (P_2) associated to $\tilde{\boldsymbol{\zeta}}^{\tilde{r}} \in Z$, there is a linear minimization program. As mentioned in remark 5.2.1, this minimization program can take two forms: $(\hat{P}_{mult})$ and $(\hat{P}_{single})$. We denote by R_Z the set of indices r related to the set of vectors Z.

With the formulation $(\hat{P}_{mult})$ for the inner minimization problem

The problem $(\hat{P}_{mult})$ has the following formulation:

$$(\hat{P}_{mult})(\boldsymbol{\zeta}^{\tilde{r}}, \tilde{\boldsymbol{d}}) \begin{cases} \min\limits_{(\boldsymbol{\varphi}, a^{\tilde{r}})} & \sum\limits_{j \in V^*} a_j^{\tilde{r}} \\ \text{s.t.} & \sum\limits_{j \in V^*} \boldsymbol{\varphi}_{ij} \leq m\boldsymbol{\zeta}_i^{\tilde{r}}, \forall i \in V^* & (5.5) \\ & \sum\limits_{i \in V^*} \boldsymbol{\varphi}_{ij} + a_j^{\tilde{r}} \geq \tilde{d}_j, \forall j \in V^* & (5.6) \\ & \boldsymbol{\varphi}_{ij} \leq \mathbf{f}_{ij}, \forall (i,j) \in V^{*2} & (5.7) \\ & a^{\tilde{r}} \geq 0, \boldsymbol{\varphi}_{ij} \in \mathbb{R}^+, \forall (i,j) \in V^{*2} \end{cases}$$

The constraint sub-matrix corresponding to variables $\boldsymbol{\varphi}_{ij}$ is a flow constraint matrix and so is totally unimodular. Since any sub-matrix corresponding to variables $a_j^{\tilde{r}}$ is a diagonal matrix, the matrix of constraints (5.5) and (5.6) is totally unimodular too. Constraints (5.7) being bound constraints, the constraint matrix of $(\hat{P}_{mult})(\tilde{\boldsymbol{\zeta}}^{\tilde{\boldsymbol{r}}}, \tilde{\boldsymbol{d}})$ is totally unimodular. So the constraints on the integrity of $\boldsymbol{\varphi}_{ij}$ variables have been replaced by $\boldsymbol{\varphi}_{ij} \in \mathbb{R}^+$ and $(\hat{P}_{mult})(\tilde{\boldsymbol{\zeta}}^{\tilde{\boldsymbol{r}}}, \tilde{\boldsymbol{d}})$ has an integer optimal solution. Moreover, $(\hat{P}_{mult})(\tilde{\boldsymbol{\zeta}}^{\tilde{\boldsymbol{r}}}, \tilde{\boldsymbol{d}})$ has a finite value since $\boldsymbol{a}^{\tilde{\boldsymbol{r}}}$ is not bounded. So we can consider its dual. Let $\boldsymbol{\lambda}_i, i \in V^*$, be the dual variables associated to constraints (5.5), $\boldsymbol{\mu}_j, j \in V^*$, the dual variables associated to constraints (5.6) and $\boldsymbol{\pi}_{ij}, (i,j) \in V^{*2}$, the dual variables associated to constraints (5.7), the dual program of $(\hat{P}_{mult})(\tilde{\boldsymbol{\zeta}}^{\tilde{\boldsymbol{r}}}, \tilde{\boldsymbol{d}})$, for given $\tilde{\boldsymbol{\zeta}}^{\tilde{r}} \in Z$, is:

$$(D\hat{P}_{mult}) \begin{cases} \max\limits_{\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\pi}} & \sum\limits_{j \in V^*} (\boldsymbol{\mu}_j \tilde{d}_j - \boldsymbol{\lambda}_j m \tilde{\zeta}_j^{\tilde{r}}) - \sum\limits_{(i,j) \in V^{*2}} \boldsymbol{\pi}_{ij} f_{ij} \\ & \boldsymbol{\mu}_j \leq 1, \, \forall j \in V^* \\ & \boldsymbol{\mu}_j - \boldsymbol{\lambda}_i - \boldsymbol{\pi}_{ij} \leq 0 \, \forall (i,j) \in V^{*2} \\ & \boldsymbol{\lambda}_i \geq 0 \, \forall i \in V^*, \boldsymbol{\mu}_j \geq 0 \, \forall j \in V^*, \boldsymbol{\pi}_{ij} \geq 0 \, \forall (i,j) \in V^{*2} \end{cases} \tag{5.8}$$

and we denote by DP_G the set of constraints of $(D\hat{P}_{mult})$.

Proposition 5.2.3. *In $(D\hat{P}_{mult})$ the constraints $0 \leq \boldsymbol{\mu}_j \leq 1, \, \forall j \in V^*$ can be replaced by $\boldsymbol{\mu}_j \in \{0,1\} : \forall j \in V^*$.*

Proof. Since the constraint matrix in $(\hat{P}_{mult})(\tilde{\boldsymbol{\zeta}}^{\tilde{\boldsymbol{r}}}, \tilde{\boldsymbol{d}})$ is totally unimodular, the constraint matrix of $(D\hat{P}_{mult})$ which is its transposed matrix is totally unimodular too and there is an optimal solution such that the variables are integer and in particular, since $\boldsymbol{\mu}_j \leq 1$, such that $\boldsymbol{\mu}_j \in \{0,1\} : \forall j \in V^*$. □

Proposition 5.2.4. *(P_2) can be modeled as a mixed integer linear program. We denote*

by (P_{mult2}) *this formulation:*

$$(P_{mult2}) \begin{cases} \max\limits_{\boldsymbol{\theta},\boldsymbol{d},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\pi},\boldsymbol{y}} & \boldsymbol{\theta} \\ s.t. & \boldsymbol{d} \in \mathfrak{D} \\ & \boldsymbol{\theta} \leq \sum\limits_{j \in V^*} (\mathbf{y}_j^r - \boldsymbol{\lambda}_j^r m \tilde{\boldsymbol{\zeta}}_j^{\tilde{r}}) - \sum\limits_{(i,j) \in V^{*2}} \boldsymbol{\pi}_{ij}^r f_{ij},\ \forall \boldsymbol{\zeta}^r \in Z \\ & \boldsymbol{\mu}_j^r \leq 1,\ \forall r \in R_Z, \forall j \in V^* \\ & \boldsymbol{\mu}_j^r - \boldsymbol{\lambda}_i^r - \boldsymbol{\pi}_{ij}^r \leq 0,\ \forall r \in R_Z, \forall (i,j) \in V^{*2} \\ & \mathbf{y}_j^r \leq \boldsymbol{\mu}_j^r d_j^{max},\ \forall r \in R_Z, \forall j \in V^* \\ & \mathbf{y}_j^r \leq d_j,\ \forall r \in R_Z, \forall j \in V^* \\ & \mathbf{y}_j^r \geq d_j - (1 - \boldsymbol{\mu}_j^r) d_j^{max},\ \forall r \in R_Z, \forall j \in V^* \\ & \boldsymbol{\lambda}_i^r \geq 0 \, \forall r, \forall i \in V^*, \boldsymbol{\mu}_j^r \in \{0,1\},\ \forall r \in R_Z, \forall j \in V^* \\ & \mathbf{y}_j^r \geq 0,\ \forall r \in R_Z, \forall j \in V^*, \boldsymbol{\pi}_{ij}^r \geq 0,\ \forall r \in R_Z, \forall (i,j) \in V^{*2} \\ & d_j \geq 0,\ \forall j \in V^* \end{cases}$$

Proof. The minimization problem within constraint (5.2) can be dualized in $(D\hat{P}_{single})$, and as (P_2) maximizes $\boldsymbol{\theta}$, the maximization can be removed. However, the dual objective function has quadratic terms $\mu_j \tilde{d}_j$. Proposition 5.2.3 allows us to linearize these products by stating $y_j^r = \mu_j^r \tilde{d}_j,\ \forall r, \forall j \in V$ by the means of the following set of constraints:

$$\begin{gathered} y_j^r \leq \boldsymbol{\mu}_j^r d_j^{max},\ \forall r \in R_Z, \forall j \in V^* \\ \mathbf{y}_j^r \leq d_j,\ \forall r \in R_Z, \forall j \in V^* \\ \mathbf{y}_j^r \geq d_j - (1 - \boldsymbol{\mu}_j^r) d_j^{max},\ \forall r \in R_Z, \forall j \in V^* \\ \mathbf{y}_j^r \geq 0,\ \forall r \in R_Z, \forall j \in V^* \end{gathered}$$

□

Remark 5.2.3. *Note that at each iteration of Algorithm 7, a set of boolean variables* $\boldsymbol{\mu}^r$ *must be added to the problem.*

In the end, we were able to propose a mixed integer linear program to solve (P_2). However, as it is stressed by remark 5.2.3, once integrated in algorithm 7, solving the problem will become more and more complex as new integer columns and constraints are added.

With the formulation $(\hat{P}_{single})$ for the inner minimization problem

Instead of using $(\hat{P}_{mult})$, let us now use the formulation $(\hat{P}_{single})$ for the inner minimization in (P_2). The same principle will be used to transform (P_2) into a mixed integer linear program. The difference here will lie in the linearization of the quadratic terms.

Proposition 5.2.5. *(P_2) can be modeled as a mixed integer linear program. We denote by ($P_{single2}$) this formulation:*

$$(P_{single2}) \begin{cases} \max\limits_{\boldsymbol{d},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\pi},\boldsymbol{y}} & \boldsymbol{\theta} \\ s.t. & \boldsymbol{d} \in \mathfrak{D} \\ & \boldsymbol{\theta} \leq \sum\limits_{j \in V^*} (\mathbf{y}_j^r - \boldsymbol{\lambda}_j^r m \tilde{\boldsymbol{\zeta}}_j^{\tilde{r}}) - \sum\limits_{(i,j) \in V^{*2}} \boldsymbol{\pi}_{ij}^r f_{ij}, \, \forall \boldsymbol{\zeta}^r \in Z \\ & \sum\limits_{j \in V^*} \boldsymbol{\mu}_j^r \leq 1, \, \forall r \\ & \boldsymbol{\mu}_j^r - \boldsymbol{\lambda}_i^r - \pi_{ij}^r \leq 0, \, \forall r, \forall (i,j) \in V^{*2} \\ & d_j = \sum\limits_{e=1}^{p_j} 2^{e-1} \mathbf{b}_j^e, \, \forall j \in V^* \\ & y_j^r = \sum\limits_{e=1}^{p_j} \mathbf{t}_j^{e,r}, \, \forall r, \forall j \in V^* \\ & \mathbf{t}_j^{e,r} \leq \mathbf{b}_j^e, \, \forall r, \forall j \in V^* \\ & \mathbf{t}_j^{e,r} \leq \boldsymbol{\mu}_j^r, \, \forall r, \forall j \in V^*, \forall e = 1, ..., p_j \\ & \mathbf{t}_j^{e,r} \geq \boldsymbol{\mu}_j^r + \mathbf{b}_j^e - 1, \, \forall r, \forall j \in V^*, \forall e = 1, ..., p_j \\ & \mathbf{t}_j^{e,r} \geq 0, \mathbf{b}_j^e \in \{0,1\}, \, \forall r, \forall j \in V^*, \forall e = 1, ..., p_j \\ & \boldsymbol{\lambda}_j^r \geq 0, \boldsymbol{\mu}_j^r \geq 0, \, \forall r \in R_Z, \forall j \in V^* \\ & \mathbf{y}_j^r \geq 0, \, \forall r \in R_Z, \forall j \in V^*, \boldsymbol{\pi}_{ij}^r \geq 0, \, \forall r \in R_Z, \forall (i,j) \in V^{*2} \\ & d_j \geq 0, \, \forall j \in V^* \end{cases}$$

with $p_j = \left\lceil \log_2 \left(d_j^{max} + 1 \right) \right\rceil, \, \forall j \in V^*$.

Proof. The minimization problem within constraint (5.3) can be dualized in ($D\hat{P}_{single}$), and as (P_2) maximizes $\boldsymbol{\theta}$, the maximization can be removed. However, the dual objective function has quadratic terms $\boldsymbol{\mu}_j \tilde{d}_j$. As $\boldsymbol{\mu}$ variables are continuous and demand variables $\boldsymbol{d}$ are integer, the quadratic terms can be linearized by using the technique that decomposes the integer variables $\boldsymbol{d}$ into their binary writing. Let us denote by $p_j = \left\lceil \log_2 \left(d_j^{max} + 1 \right) \right\rceil, \, \forall j \in V^*$ the number of boolean variables necessary to write d_j. For each product $\boldsymbol{\mu}_j \tilde{d}_j$, let us denote by $\mathbf{b}_j^e$ the boolean variables used to decompose $\tilde{d}_j$, for $e = 1, ..., p_j$, and by $\mathbf{t}_j^{r,e}$ the set of required continuous variables. As shown in [16],

these products can thus be linearized by the set of following constraints:

$$d_j = \sum_{e=1}^{p_j} 2^{e-1}\mathbf{b}_j^e,\ \forall j \in V^*$$
$$y_j^r = \sum_{e=1}^{p_j} 2^{e-1}\mathbf{t}_j^{e,r},\ \forall r, \forall j \in V^*$$
$$\mathbf{t}_j^{e,r} \leq \mathbf{b}_j^e,\ \forall r, \forall j \in V^*$$
$$\mathbf{t}_j^{e,r} \leq \boldsymbol{\mu}_j^r,\ \forall r, \forall j \in V^*, \forall e = 1, ..., p_j$$
$$\mathbf{t}_j^{e,r} \geq \boldsymbol{\mu}_j^r + \mathbf{b}_j^e - 1,\ \forall r, \forall j \in V^*, \forall e = 1, ..., p_j$$

□

Remark 5.2.4. *Note that at each iteration of Algorithm 7, there are no integer variables to be added if formulation ($P_{single2}$) is chosen. However, lots of boolean variables $\mathbf{b}_j^e$ must be added before the first iteration.*

From remarks 5.2.3 and 5.2.4, one can expect that both formulations (P_{mult2}) and ($P_{single2}$) have their pros and cons. With (P_{mult2}), Algorithm 7 will add integer columns at each step, thus making the problem more and more difficult, probably faster than with formulation ($P_{single2}$). However, that last formulation needs a lot of boolean variables from the beginning in order to decompose each demand variable into its binary writing. Therefore, finding the best option is not obvious, and the choice will be made on an empirical basis.

5.2.3 Experimental study of the algorithm behaviour and formulation efficiency comparison

In this section, we will proceed to a few numerical tests of our exact approach. As it was introduced in sections 5.1 and 5.2.1, we will use a combination of Algorithms 6 and 7 to solve (PON_{rob}) to optimality, Algorithm 7 being called by Algorithm 6 to solve $Q(\tilde{\mathbf{f}}, \tilde{\mathbf{z}})$. In section 5.2.1, we showed that two formulations were possible because they proposed different linearization of quadratic terms that occur in both cases. Thus, we have two main objectives for these first tests:

- Conduct a first experimental study to assess the solving potential of the approach
- Compare both formulations

First, Table 5.1 shows the details of the 6 randomly generated instances that were solved. For each instance, its number of node is given, along with its average demand per node (column "Avg demand"). Then, the optimal fiber cost and splitter cost are given

for (PON_{det}) and (PON_{rob}). Note that for all instances, the chosen value for $\bar{d}$ equaled 25% of the total demand, that is $\bar{d} = \left\lceil 0.25 \sum_{j \in V^*} d_j^{max} \right\rceil$.

The main observation is that the total cost of (PON_{rob}) is lower (the gain is between 14.2% and 32.3% depending on the instance), even though its fiber cost is always higher or equal. However, since we are only dealing with small graphs, the impact of demand limitation may be higher on larger graphs.

Instances			(PON_{det})		(PON_{rob})	
i	$\|V\|$	Avg demand	Fiber cost	Splitter cost	Fiber cost	Worst splitter cost
1	5	9.75	257	1000	278	800
2	5	16.75	680	1800	680	1000
3	5	13.75	588	1400	598	1000
4	5	18.25	1717	2000	1762	1000
5	5	14	1212	1600	1404	800
6	5	16.15	1304	1800	1353	800

Table 5.1: Robust counterpart gains compared to a deterministic approach

The reason why only 5 nodes instances were solved is partly given by Table 5.2 which compares the solving performances of (PON_{rob}) depending on the formulation used for the recourse problem that is either (P_{mult2}) or ($P_{single2}$). Note that while dealing with 5 nodes instances, the solving time oscillates between 9 and 117 seconds for (P_{mult2}), and between 5 and 238 seconds for ($P_{single2}$). This denotes the high instability of the algorithm. Note that many tests were performed with that algorithm and that for some 7 nodes instance, the algorithm could not converge in more than 10 hours. On the contrary, to illustrate further more that instability, we managed (in some rare cases) to solve instances of size 15 in less than half an hour. It emphasizes the fact that some instances seem to be quite easy to solve in practice, and some others need a huge amount of scenarios to be solved to optimality (both in the master and in the slave problems).

For each formulation, the number of generated scenarios in the master problem is reported, and the average number of generated scenarios for solving the slave problem is also reported. It is no surprise to note that the number of iteration increases with the solving time.

Both formulations have their pros and cons. On instances that seem easy to solve (that is instances 1, 3 and 5), formulation ($P_{single2}$) seems to be more efficient. However, it tends to be more unstable for harder instances whereas (P_{mult2}) performs a lot better on these ones. That lower, yet non negligible, instability is the reason why we focused on using (P_{mult2}) instead of ($P_{single2}$) for the rest of this chapter.

Instances	With (P_{mult2})			With ($P_{single2}$)		
	CPU time	Master	Avg Slave	CPU time	Master	Avg Slave
1	20	24	10.2	15	23	9.95
2	117	18	35.69	238	17	35.75
3	9	11	14.6	5	10	20.25
4	69	11	36.83	227	12	35.43
5	21	15	15.67	12	14	16.12
6	99	18	30.79	158	17	31.77

Table 5.2: Performance comparison between (P_{mult2}) and ($P_{single2}$) as recourse formulation for solving (PON_{rob})

5.3 Improving the solving procedure

Regarding results of section 5.2.3, our solving approach needs improvement in order to tackle larger instances, or at least to solve them faster. Hence, we decided to improve the algorithm by using methods from the literature to tackle instability in Benders decomposition related approaches like ours. Many were tried, conceived and tested, but we only present the 2 most relevant we found. We consider them as relevant because they can be generalized to other problems. For each potential improvement, we shall present the theoretical principle behind it, and an experimental study that assesses, or not, the effectiveness of the improvement.

5.3.1 A discriminating choice method for picking several good integer vectors $\boldsymbol{\zeta}^r \in Z$ in order to solve the recourse problem faster

Theoretical developments on the method

This method was inspired by some of the work of Magnanti and Wong [37]. At a given step k, in algorithm 7, once (P_2) has been solved for a given subset Z_2 of Z, there is a current optimal solution $\boldsymbol{d}_k^*$. Then, we will solve $R(\tilde{\mathbf{f}}, \tilde{\mathbf{z}}, \boldsymbol{d}_k^*)$ in order to get a valid integer vector $\boldsymbol{\zeta}^r$ that will be added to the description of (P_2). However, several vectors $\boldsymbol{\zeta}^r$ may be optimal solution to R. Therefore, we aim to design a discriminating method to choose the best vectors for solving (P_2) amongst the optimal solutions of R for a given $\boldsymbol{d}_k^*$. Let us denote by $Z(\boldsymbol{d}_k^*)$ the subset of Z such that all $\boldsymbol{\zeta}^r \in Z(\boldsymbol{d}_k^*)$ are optimal solutions of $R(\tilde{\mathbf{f}}, \tilde{\mathbf{z}}, \boldsymbol{d}_k^*)$. Let us recall the formulation of (Pm_2) at a given step k:

$$(P_{mult2})\begin{cases}\max\limits_{\boldsymbol{\theta},\boldsymbol{d},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\pi},\boldsymbol{y}} & \boldsymbol{\theta}\\ \text{s.t.} & \boldsymbol{d}\in\mathfrak{D}\\ & \boldsymbol{\theta}\leq\sum\limits_{j\in V}(\mathbf{y}_j^r-\boldsymbol{\lambda}_j^r m\tilde{\boldsymbol{\zeta}}_j^{\tilde{r}})-\sum\limits_{(i,j)\in V^2}\boldsymbol{\pi}_{ij}^r f_{ij},\ \forall\boldsymbol{\zeta}^r\in Z_2 \qquad (5.9)\\ & \boldsymbol{\mu}_j^r\leq 1,\ \forall r\in R_{Z_2},\forall j\in V\\ & \boldsymbol{\mu}_j^r-\boldsymbol{\lambda}_i^r-\boldsymbol{\pi}_{ij}^r\leq 0,\ \forall r\in R_{Z_2},\forall(i,j)\in V^2\\ & \mathbf{y}_j^r\leq\boldsymbol{\mu}_j^r d_j^{max},\ \forall r\in R_{Z_2},\forall j\in V\\ & \mathbf{y}_j^r\leq d_j,\ \forall r\in R_{Z_2},\forall j\in V\\ & \mathbf{y}_j^r\geq d_j-(1-\boldsymbol{\mu}_j^r)d_j^{max},\ \forall r\in R_{Z_2},\forall j\in V\\ & \boldsymbol{\lambda}_i^r\geq 0\,\forall r,\forall i\in V,\boldsymbol{\mu}_j^r\in\{0,1\},\ \forall r\in R_{Z_2},\forall j\in V\\ & \mathbf{y}_j^r\geq 0,\ \forall r\in R_{Z_2},\forall j\in V,\boldsymbol{\pi}_{ij}^r\geq 0,\ \forall r\in R_{Z_2},\forall(i,j)\in V^2\\ & d_j\geq 0,\ \forall j\in V\end{cases}$$

At step k, a new integer vector from $Z(\boldsymbol{d}_k^*)$ must be added. Looking at constraint (5.9), one would prefer to add an integer vector that can reduce $\boldsymbol{\theta}$, if possible, at the next iteration. That is, adding the vector that verifies:

$$\min_{\boldsymbol{\zeta}^r\in Z(\boldsymbol{d}_k^*)}\sum_{j\in V}(\mathbf{y}_{j,0}^r-\boldsymbol{\lambda}_{j,0}^r m\tilde{\boldsymbol{\zeta}}_j^r)-\sum_{(i,j)\in V^2}\boldsymbol{\pi}_{ij,0}^r f_{ij}$$

with $(\mathbf{y}_0^r,\boldsymbol{\lambda}_0^r,\boldsymbol{\pi}_0^r)$ being an interior point of the dual variables polyhedron, which is strictly equivalent to:

$$\max_{\boldsymbol{\zeta}^r\in Z(\boldsymbol{d}_k^*)}\sum_{j\in V}\boldsymbol{\lambda}_{j,0}^r m\tilde{\boldsymbol{\zeta}}_j^r$$

From these facts, one can derive the following proposition:

Proposition 5.3.1. *In Algorithm 7, at a given step k, once $R(\tilde{\mathbf{f}},\tilde{\mathbf{z}},\boldsymbol{d}_k^*)$ is solved, it is possible to find other valid vectors $\boldsymbol{\zeta}^r\in Z(\boldsymbol{d}_k^*)$ by solving the following mixed integer linear program (PZ^k):*

$$(PZ^k)\begin{cases}\max & \sum\limits_{j\in V}\boldsymbol{\lambda}_{j,0}m\tilde{\boldsymbol{\zeta}}_j^r\\ s.t. & c\boldsymbol{\zeta}^r=v\left(R(\tilde{\mathbf{f}},\tilde{\mathbf{z}},\boldsymbol{d}_k^*)\right)\\ & (\boldsymbol{\varphi},\boldsymbol{\zeta}^r)\in P_G(\boldsymbol{d}_k^*)\\ & \boldsymbol{\zeta}^r\leq\tilde{\boldsymbol{z}}\\ & \boldsymbol{\varphi}\leq\tilde{\mathbf{f}}\\ & \boldsymbol{\zeta}^r\in\mathbb{N}^{|V|},\ \boldsymbol{\varphi}\geq 0\end{cases}$$

for any vector $\boldsymbol{\lambda}_0\in\mathbb{R}_{*+}^{|V|}$.

Proof. Constraints of (PZ^k) ensures that its solution is such that it belongs to the set of solutions of $R(\tilde{\mathbf{f}}, \tilde{\mathbf{z}}, \boldsymbol{d}_k^*)$ and that its value equals $v\left(R(\tilde{\mathbf{f}}, \tilde{\mathbf{z}}, \boldsymbol{d}_k^*)\right)$. Therefore, any vector that verifies these constraints already belongs to $Z(\boldsymbol{d}_k^*)$. As mentioned before, by maximizing $\sum_{j \in V} \boldsymbol{\lambda}_{j,0}^r m \tilde{\boldsymbol{\zeta}}_j^r$ with these constraints, for $\boldsymbol{\lambda}_0$ being an interior point, we aim to find a "good" vector. An interior point corresponds to strictly positive values of $\boldsymbol{\lambda}_0$ coefficients (see proof of Proposition 5.2.4). □

Remark 5.3.1. *Note that by picking different $\boldsymbol{\lambda}_0 \in \mathbb{R}_{*+}^{|V|}$, solving (PZ^k) may lead to different optimal useful $\boldsymbol{\zeta}^r$ vectors.*

From Proposition 5.3.1, we propose a slight variant of Algorithm 7, called Algorithm 8, that allows one to potentially find better $\boldsymbol{\zeta}^r$ vectors to add to Z_2 at each step.

Algorithm 8 A column-and-constraint generation algorithm variant for solving $Q(\tilde{\mathbf{z}}, \tilde{\mathbf{f}})$ to optimality

Require: $UB = +\infty$, $\varepsilon \in \mathbb{R}_+$, $k = 0$;

find an initial vector $\boldsymbol{\zeta}^1$ by solving (5.3) for a randomly chosen scenario $\boldsymbol{d} \in \mathfrak{D}$
$Z_2 \leftarrow \left\{\boldsymbol{\zeta}^1\right\}$
while $UB > \varepsilon$ **do**
  solve (P_2) with Z_2; let $\boldsymbol{d}_k^*$ be an optimal solution;
  $UB \leftarrow v(P_2)$;
  solve (5.3);
  solve (PZ^k); let $\boldsymbol{\zeta}^{k+1}$ be an optimal solution;
  $Z_2 \leftarrow Z_2 \cup \{\boldsymbol{\zeta}^{k+1}\}$; $k \leftarrow k + 1$;
end while
return $\max_{k' \in \{1,\dots,k\}} c\boldsymbol{\zeta}^{k'}$ and the associated $\boldsymbol{d}_{k'}$ with $k^* = \arg \max_{k' \in \{1,\dots,k\}} c\boldsymbol{\zeta}^{k'}$;

Experimental testing of Algorithm 8

Algorithm 8 was extensively tested on several instances with very different λ_0 configurations. Unfortunately, the outcome was almost everytime the same: the solution found by solving (PZ^k) is the same as the one found by solving (5.3). To us, this is due to the fact that we are only working on very small graphs, thus making the number of equivalent solutions that could be found by solving (PZ^k) very limited in practice. Note that this is not due to a particular property of the problem since it happened, yet rarely, that the solution of (PZ^k) was different. However, since it was so rare and unpredictable, its impact was hard to quantify.

However, we think that on larger graphs, this approach would have a higher impact that we can only forecast from our test basis.

5.3.2 Avoiding potential instability by using Ordered Weighted Average as the objective function of the recourse problem

Theoretical developments on the method

While solving the recourse problem by the means of Algorithm 7, let us consider the problem (P_{mult2}) at a given step k of the algorithm. There are k constraints (5.9), each one limiting the value of variable $\boldsymbol{\theta}$. Let us denote by $v(r)$ the value of the right-hand-side of a line r (among constraints (5.9)). Let us assume these lines sorted in an increasing order $r_1, r_2, ..., r_k$ with $val(r_1) \leq val(r_2) \leq ... \leq val(r_k)$. Hence, the optimal solution of (P_{mult2}) is the one that maximizes the value of line r_1. Therefore, that optimal solution does not take into account $val(r_2)$, nor $val(r_3)$, nor any other. However, it is not impossible that at the next iteration $k+1$, on of those lines become the minimal one, thus making the next optimal solution computed from that line. Then, it could be interesting, in order to avoid instability of successive (P_{mult2}) solutions, to somehow anticipate by taking into account at each iteration these 2, 3 or more minimal lines in the optimization.

In order to do so, it is possible to consider the Ordered Weighted Average operator (OWA), described in the section 1.3.2 of the Bibliographical Study to which the reader is referred. As OWA aggregates several criteria, let us define those we will work with. A criterion is defined as such:

$$v_r(\boldsymbol{d}, \boldsymbol{\zeta}^r) = \min_{\boldsymbol{\varphi} \text{ s.t.} (\boldsymbol{\varphi}, a^r) \in \hat{P}_G(\boldsymbol{d}, \boldsymbol{\zeta}^r)} \sum_{j \in V} a_j^r, \ \forall r = 1, ..., R$$

where $R = |Z|$. In addition, let us consider a set of positive weights $\delta_1 \geq \delta_2 \geq ... \geq \delta_q > \delta_{q+1} = ... = \delta_R = 0$, with $q < R$ and $\sum_{i=1}^{R} \delta_i = 1$. As in section 1.3.2, we define the bijection from $\{1, ..., R\}$ to $\{1, ..., R\}$ that sorts criteria in an increasing order: $v_{(1)} \leq ... \leq v_{(R)}$. We consider a subset of indexes of $k \geq q$ criteria $K \subset \{1, ..., R\}$ with $|K| = k$, and we define the bijection from $\{1, ..., k\}$ to K that sorts selected criteria by K in an increasing order: $v_{[1]} \leq ... \leq v_{[k]}$.

Proposition 5.3.2.

$$0 \leq v_{(1)}(\boldsymbol{d}, \boldsymbol{\zeta}^{(1)}) \leq \sum_{r=1}^{R} \delta_r v_{(r)}(\boldsymbol{d}, \boldsymbol{\zeta}^{(r)}) = \sum_{r=1}^{q} \delta_r v_{(r)}(\boldsymbol{d}, \boldsymbol{\zeta}^{(r)}) \leq \sum_{r=1}^{q} \delta_r v_{[r]}(\boldsymbol{d}, \boldsymbol{\zeta}^{[r]}) = \sum_{r=1}^{k} \delta_r v_{[r]}(\boldsymbol{d}, \boldsymbol{\zeta}^{[r]})$$

Proof. The first term is the smallest one. The second one is the ordered weighted average of the q smallest criteria of $\{1, ..., R\}$. The third one is the ordered weighted average of q criteria of $\{1, ..., R\}$ that may not be the smallest ones. □

We will now describe the variant of Algorithm 7 that uses the OWA aggregation operator, and we will prove that they both converge towards the same objective. First, we still have a set of weights $\delta_1 \geq \delta_2 \geq ... \geq \delta_q > \delta_{q+1} = ... = \delta_R = 0$ with $q < R$ and $\sum_{i=1}^{R} \delta_i = 1$ as mentioned above. We set $k = q$ and $\boldsymbol{\zeta}^1, ..., \boldsymbol{\zeta}^k$ chosen in Z. We denote by (P^{OWA}_{mult2}) the following mixed integer program:

$$(P^{OWA}_{mult2}) \begin{cases} \max\limits_{\boldsymbol{v},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{d},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\pi},\boldsymbol{y}} & \sum\limits_{r=1}^{k} \boldsymbol{\alpha}_r + \boldsymbol{\beta}_r \\ \text{s.t.} & \boldsymbol{d} \in \mathfrak{D} \\ & \boldsymbol{\alpha}_r + \boldsymbol{\beta}_{r'} \leq \delta_{r'} \mathbf{v}_r, \, \forall r, r' = 1, ..., k \qquad (5.10) \\ & \mathbf{v}_r \leq \sum\limits_{j \in V} (\mathbf{y}^r_j - \boldsymbol{\lambda}^r_j m \boldsymbol{\zeta}^r_j) - \sum\limits_{(i,j) \in V^2} \boldsymbol{\pi}^r_{ij} f_{ij}, \, \forall r = 1, ..., k \qquad (5.11) \\ & \boldsymbol{\mu}^r_j \leq 1, \, \forall r = 1, ..., k, \forall j \in V \\ & \boldsymbol{\mu}^r_j - \boldsymbol{\lambda}^r_i - \boldsymbol{\pi}^r_{ij} \leq 0, \, \forall r = 1, ..., k, \forall (i,j) \in V^2 \\ & \mathbf{y}^r_j \leq \boldsymbol{\mu}^r_j d^{max}_j, \, \forall r = 1, ..., k, \forall j \in V \\ & \mathbf{y}^r_j \leq d_j, \, \forall r = 1, ..., k, \forall j \in V \\ & \mathbf{y}^r_j \geq d_j - (1 - \boldsymbol{\mu}^r_j) d^{max}_j, \, \forall r = 1, ..., k, \forall j \in V \\ & \boldsymbol{\lambda}^r_i \geq 0 \, \forall r = 1, ..., k, \forall i \in V \\ & \boldsymbol{\mu}^r_j \in \{0, 1\}, \, \forall r = 1, ..., k, \forall j \in V \\ & \mathbf{y}^r_j \geq 0, \, \forall r = 1, ..., k, \forall j \in V \\ & \boldsymbol{\pi}^r_{ij} \geq 0, \, \forall r = 1, ..., k, \forall (i,j) \in V^2 \\ & d_j \geq 0, \, \forall j \in V \end{cases}$$

This problem minimizes the OWA of the above defined criteria (see section 1.3.2 for more details).

Before entering the details, let us prove the following proposition.

Proposition 5.3.3. *If $v(P^{OWA}_{mult2}) = 0$, then the recourse problem is solved.*

Proof. Let $\boldsymbol{d}$ be the solution of the last iteration k. From Proposition 5.3.2, we have:

$$0 \leq \min_{r=1,...,R} v_r(\boldsymbol{d}, \boldsymbol{\zeta}^r) \leq \sum_{r=1}^{k} \delta_r v_{[r]}(\boldsymbol{d}, \boldsymbol{\zeta}^{[r]})$$

Where [.] is the bijection from $\{1, ..., k\}$ to $K \subset \{1, ..., R\}$ the set of indexes of the k criteria chosen by the algorithm such that $v_{[1]} \leq ... \leq v_{[k]}$. The last term equals 0 since it is the objective function of the current problem. □

Let $\theta = \delta_1 \min_{r=1,...,k} \mathbf{v}_r$ be the first term of the OWA. Let us examine the different cases.

If $\theta > 0$, then let $\boldsymbol{d}^{k+1}$ be the solution of (P^{OWA}_{mult2}). We then solve $R(\tilde{\mathbf{f}}, \tilde{\mathbf{z}}, \boldsymbol{d}^k)$ in order to obtain the solution $\boldsymbol{\zeta}^{k+1}$. Then we need to add the columns and constraints related to that new integer vector to problem (P^{OWA}_{mult2}), we set $k = k + 1$ in order to solve it again. Thus, we add a set of constraints that are violated by the current solution.

On the contrary, if $\theta = 0$, there are two cases. If OWA also equals 0 (i.e. $v(P^{OWA}_{mult2}) = 0$), Proposition 5.3.3 implies that the recourse problem has been solved. If not, one needs to change the sequence of weights by putting left its terms as follows:

$$\delta_1 = \delta_1 + \delta_2,\ \delta_2 = \delta_3, ...,\ \delta_q = \delta_{q+1}$$

and then solve (P^{OWA}_{mult2}) again. Proposition 5.3.4 proves that this process converges.

Proposition 5.3.4. *Assuming that all $\delta_i > 0$ weights are different for $i = 1$ to q and let us suppose $k \geq q$. If, after the choice made at the end of iteration k was to put the δ sequence to the left, the solution at the next iteration $k + 1$ of problem (P^{OWA}_{mult2}) is unchanged from iteration k, or if criteria v_r $(r = 1, ..., k)$ do not change values, then the objective function of the recourse problem strictly decreases.*

Proof. When the weights sequence is put to the left, it is because the first criterion equals 0 (i.e. the smallest one), and that $OWA \neq 0$. It implies that $q > k'$, k' being the rank of the last null criterion in OWA. By putting the sequence to the left, we obtain a new sequence such that $\delta'_i = \delta_{i+1}$ for $i \geq 2$. Both sequences satisfies the conditions of Property 1.3.2. The result thus follows from that Property. □

Hence, we designed a variant of Algorithm 7 that tries to avoid instability problems by using the aggregation operator OWA. However, many issues remain, such as the choice of δ values for instance. And, more importantly, the effectiveness of this variant remains to be proven. Before doing so, let us sum up what has been done so far by describing formally that variant under the name of Algorithm 9.

Experimental testing of Algorithm 9

In order to illustrate how the Algorithm 9 may improve the solving of problem (PON_{rob}), we ran the algorithm for a single instance first, in order to find good δ coefficients. Two sets of coefficients are presented in these tests.

- $\delta^1_1 = 0.5 + \Delta^1,\ \delta^1_2 = 0.5 - \Delta^1$ for $\Delta^1 \in \{0, 0.05, 0.1, ..., 0.45\}$
- $\delta^2_1 = 0.4 + \Delta^2,\ \delta^2_2 = 0.3 - \dfrac{\Delta^2}{2},\ \delta^2_3 = 0.3 - \dfrac{\Delta^2}{2}$ for $\Delta^2 \in \{0, 0.04, 0.08, ..., 0.56\}$

Algorithm 9 A column-and-constraint generation algorithm variant based on the OWA aggregation operator for solving $Q(\tilde{\mathbf{z}}, \tilde{\mathbf{f}})$ to optimality

Require: $UB = +\infty$, $\varepsilon \in \mathbb{R}_+$, $k = 0$;
Require: $\delta_1 \geq \delta_2 \geq ... \geq \delta_q$
find an initial vector $\boldsymbol{\zeta}^1$ by solving (5.3) for a randomly chosen scenario $\boldsymbol{d} \in \mathfrak{D}$
$Z_2 \leftarrow \left\{\boldsymbol{\zeta}^1\right\}$
while $UB > \varepsilon$ **do**
  solve (P^{OWA}_{mult2}) with Z_2; let $\boldsymbol{d}^*_k$ be an optimal solution;
  $UB \leftarrow v(P^{OWA}_{mult2})$;
  $\theta \leftarrow \delta_1 \min_{r=1,...,k} \mathbf{v}_r$
  if $\theta > 0$ **then**
    solve (5.3); let $\boldsymbol{\zeta}^{k+1}$ be an optimal solution;
    $Z_2 \leftarrow Z_2 \cup \{\boldsymbol{\zeta}^{k+1}\}$; $k \leftarrow k + 1$;
  else
    if $UB > 0$ **then**
      $\delta_1 \leftarrow \delta_1 + \delta_2$, $\delta_2 \leftarrow \delta_3$, ..., $\delta_q \leftarrow \delta_{q+1}$
    end if
  end if
end while
return $\max_{k' \in \{1,...,k\}} c\boldsymbol{\zeta}^{k'}$ and the associated $\boldsymbol{d}_{k'}$ with $k^* = \arg\max_{k' \in \{1,...,k\}} c\boldsymbol{\zeta}^{k'}$;

Note that others were tried. We pushed tests to $q = 5$ but the behavior presented below was always observed. The instance we are dealing with is a randomly generated instance of size 7 that is solved in 32 seconds by the standard algorithm 6. For each set of coefficient, we solve the problem for all Δ^k values. For every problem solved, we report the gain in solving time that we observe.

Figures 5.1 and 5.2 shows the results for both sets of coefficients. In both cases, no trend can be identified and the algorithm performance seem to be very sensitive to Δ^k variation. From there, we took for each set of coefficient the ones that performed best on our instance:

- $\delta^1_1 = 0.8$ and $\delta^1_2 = 0.2$.
- $\delta^2_1 = 0.56$, $\delta^2_2 = 0.22$ and $\delta^2_3 = 0.22$.

We then used these coefficients in order to solve 10 other instances of similar size that were previously solved by algorithm 6. For each instance, we report the gain in solving time for both set of coefficient. Results are reported in Figure 5.3. There again, no trend can be identified and, more importantly, both curves have no common trend at all.

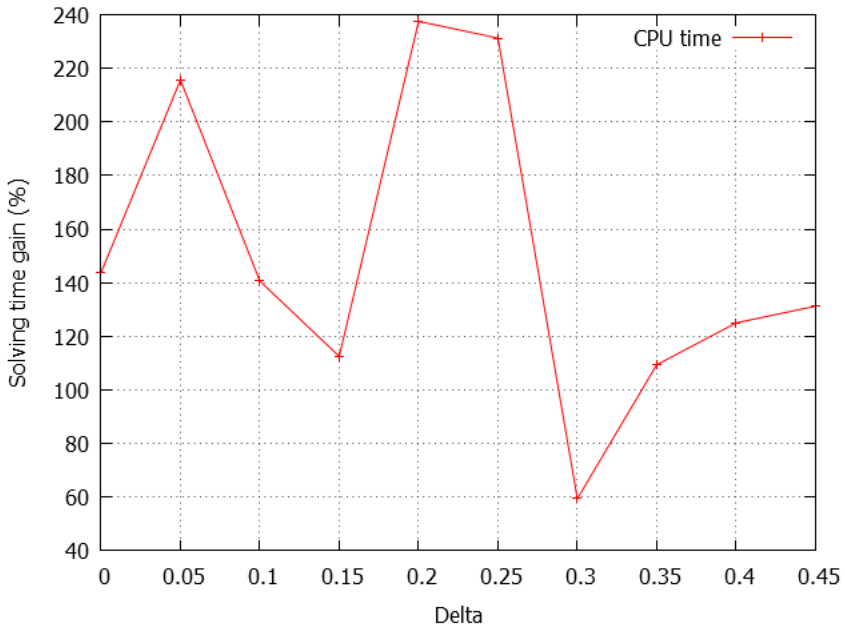

Figure 5.1: Solving time gains of Algorithm 9 for the 2 δ^1 coefficients

Moreover, the solving time is often longer, which means that the "best" set of coefficient for an instance may not be the best for another.

In the end, we showed that algorithm 9 can perform better with the right set of coefficient, but no rule could be identified on how to pick good coefficient since a set of coefficient can improve the solving time on an instance can also deteriorate it a lot on another.

5.4 Study of what makes "robust" a solution

In this section, let us examine a robust solution on a small instance we were able to solve to optimality. This instance is presented on Figure 5.4 where numbers on grey arcs denotes the routing cost of a single fiber, and numbers on orange arcs denotes the routing cost of a fiber from the OLT and the splitter installation cost. Note that the splitter installation cost alone equals 138, and its capacity equals 8. Finally, demands are shown on the right-hand-side nodes. The total demand equals 35.

Let us first examine what the exact solution of the deterministic problem would be, as it is presented on Figure 5.5. On this solution, 5 splitters are installed (for a total cost of 690), and fibers are used according to the demand for a total cost of 115. Note that the amount of fiber incoming a demand strictly equals the demand of that node so that

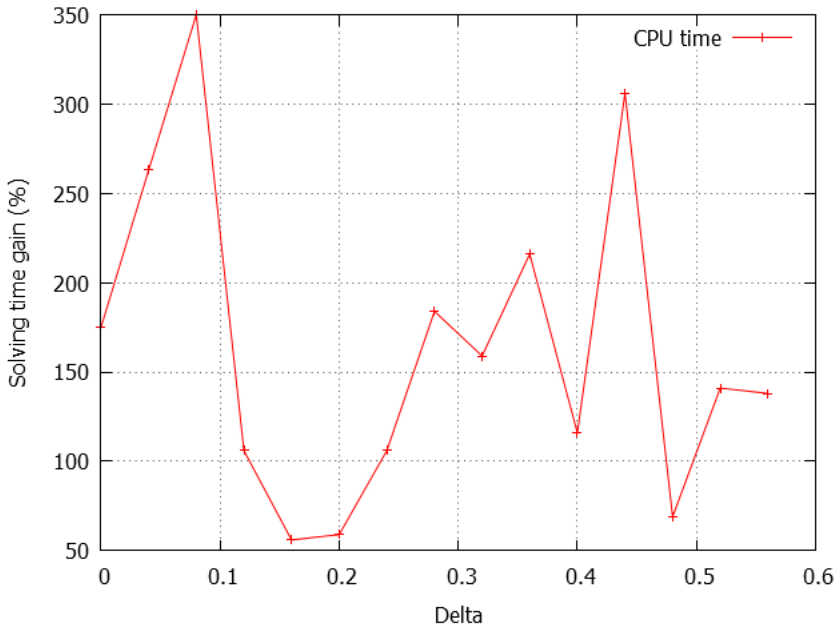

Figure 5.2: Solving time gains of Algorithm 9 for the 3 δ^2 coefficients

no fiber is unused. On the third node from the top, 2 splitters are used for the third and fourth nodes.

Let us now take a look at the optimal solution of (PON_{rob}), shown on Figure 5.6, for a total expected demand value of 11. Only first stage variables values are shown on this figure. The fiber cost equals 197, which is higher than for the deterministic version of the problem. On the contrary, the worst splitter cost possible equals 414 (which is equivalent to 3 splitters). On the right of the figure, the first scenarios added in the master problem by the Algorithm 6 are shown. The most important thing to note is the fact that the incoming number of fibers on the second and fourth node are greater than the maximal demand value of these nodes. For the second node, 7 fibers are incoming whereas the maximum demand possible on that node equals 6. On the fourth node, 15 fibers are incoming for a maximum demand of 11. That explains the higher fiber cost for that robust solution. The point of having more fibers incoming in a robust context is to enable more routing options for second stage fiber variables, thus making more localization options for the second stage splitter variables. This is how the worst case splitter cost has been reduced. It is also important to note that fibers that may be considered as "additional" are all coming from the third node. That grouping trend was observed almost every time while solving (PON_{rob}).

To sum up what this small example brought, there are two main orientations that we identified that can make a solution robust, regarding demand uncertainty in a PON

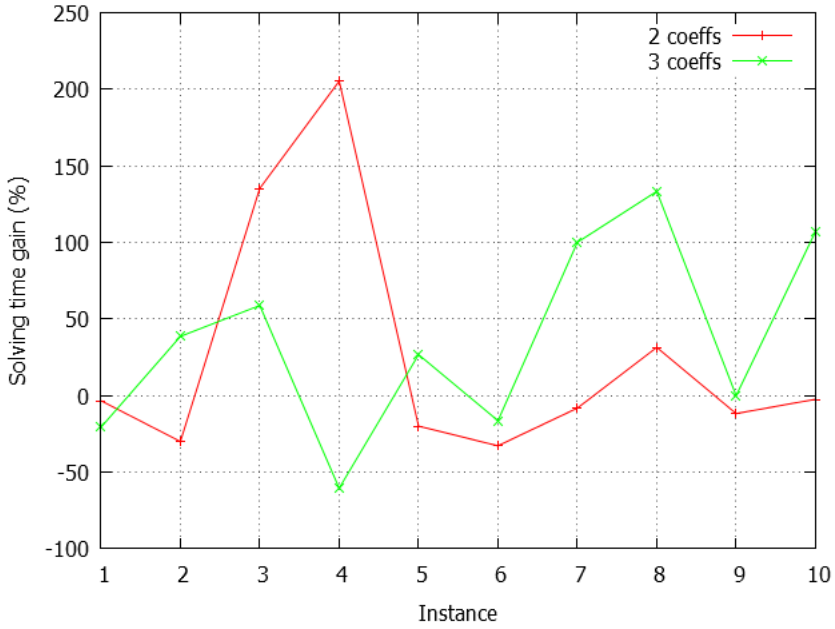

Figure 5.3: Solving time gains of Algorithm 9 for the 3 δ^2 coefficients

deployment context:

- Having more fibers than required for the full demand scenario.
- Having nodes that regroup lots of potential splitters.

Conclusion

In this section, we engaged several attempts to solve (PON_{rob}) to optimality. A column-and-constraint generation algorithm that solves its slave problem by the means of another column-and-constraint generation algorithm was designed and tested. Unfortunately, its behavior is highly unstable, and multiple stabilization techniques were employed. However, their impact hard to quantify and to master. Hence, the instance size we were able to solve with this approach remains limited. Yet we think that some of the stabilization methods we tried may be useful for other problems. However, solving the problem to optimality enabled us to extract information on what makes a solution robust.

At this point, we made the choice to let the exact solving of the problem aside as our reflection on the problem led us to think that more interesting options were to be tried regarding non-exact methods.

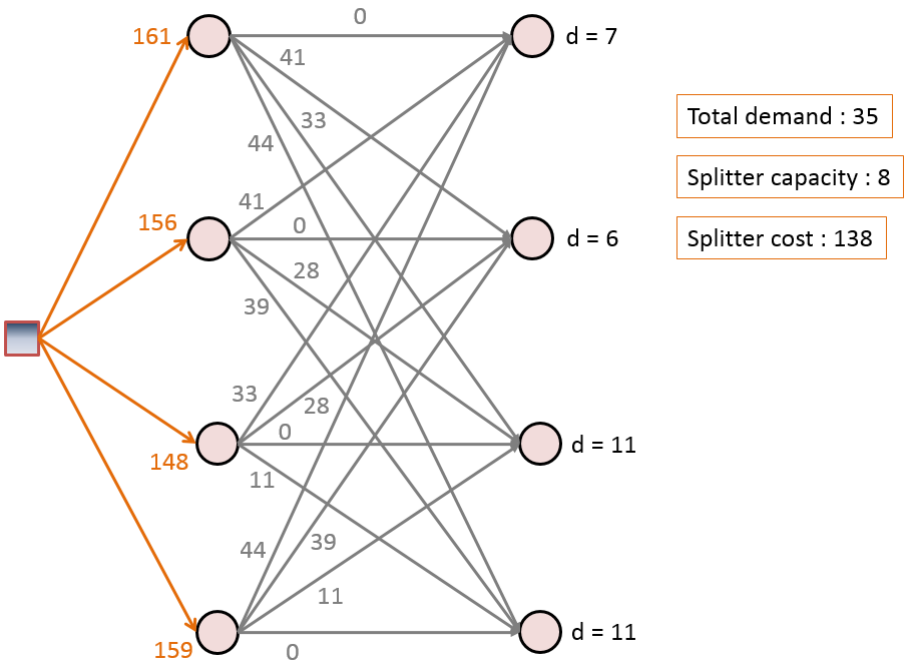

Figure 5.4: Description of an instance with its costs and its demands

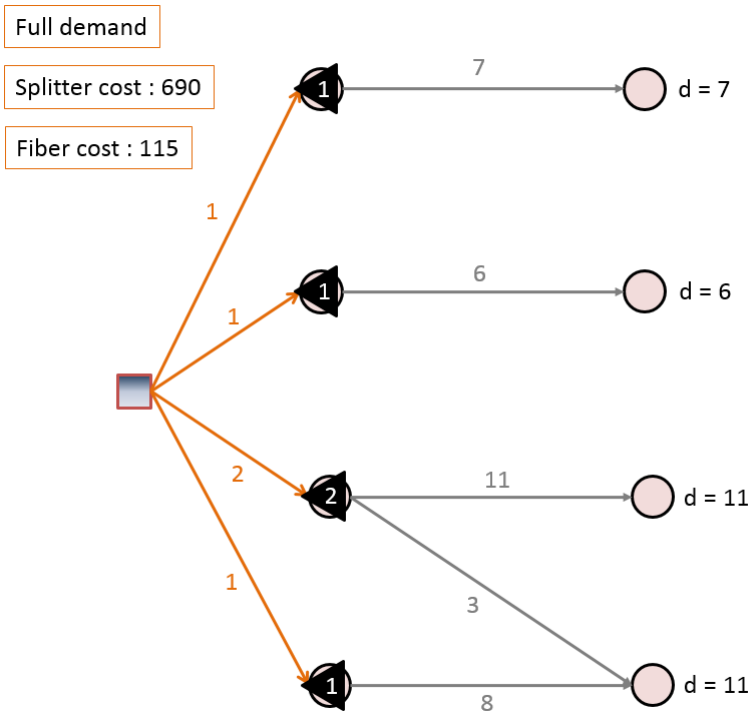

Figure 5.5: Optimal solution of (PON_{det})

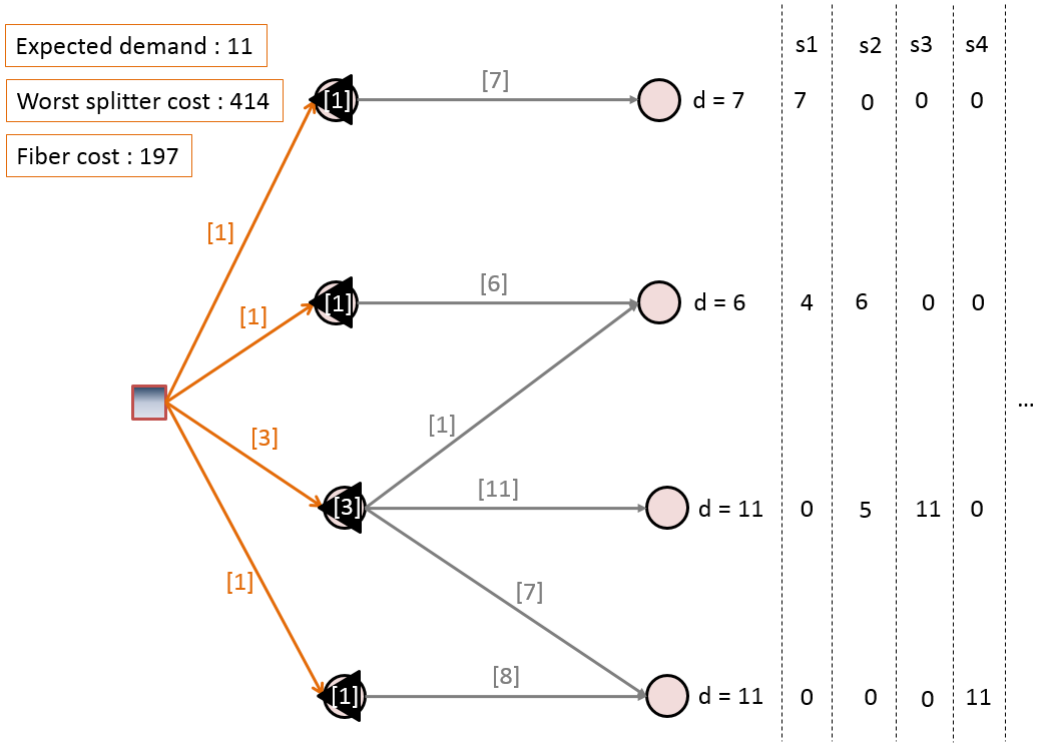

Figure 5.6: Optimal solution of (PON_{rob})

Chapter 6

Development of non exact solving approaches for the two-stage robust passive optical network design problem under demand uncertainty

Introduction

Chapter 5 focused on solving the robust problem to optimality. As the range of problems that one is able to tackle with these algorithms is too limited for real-life instances, it is thus important to develop solving methods that can deal with larger problem sizes. Of course, for tackling this problem, many methods are available in the literature but none have yet been applied to the specific problem of the two-stage robust passive optical network deployment optimization problem.

Thus, many options are available to us. As shown in the bibliographical study of section 1.2, many methods are available to solve two-stage robust problems with a continuous recourse. These methods could then be applied to our problem, at the cost of a continuous relaxation. But it is not the only way, as the previous chapter allowed us to gain insight on what makes a solution robust. From that information, it may be possible to design specific heuristics to solve the problem. As one can expect, many issues will arise, and as nothing exists yet for this problem, it is important to proceed to the exploration of our solving options.

The main objective of this chapter is to design and compare several non exact solving approaches for the two-stage robust passive optical network design problem under demand uncertainty. All methods will thus have to be evaluated in order to conclude on their performance, or their interest.

6.1 Design of a heuristic for solving the recourse problem

The robust problem has proven difficult to solve. Thus, we aim to find a good heuristic for solving the recourse problem (4.16). Let us recall it:

$$Q(\mathbf{f},\mathbf{z}) \ : \max_{d\in\mathfrak{D}} \ \min_{\substack{(\boldsymbol{\varphi},\boldsymbol{\zeta})\in P_G(d)\\ \boldsymbol{\zeta}\leq\mathbf{z},\ \boldsymbol{\varphi}\leq\mathbf{f}}} \ \sum_{i\in V^*} C\boldsymbol{\zeta}_i$$

6.1.1 Definitions and principles prior to the design of the heuristic

First, let us point out some properties of the recourse problem.

Proposition 6.1.1. *There always exists an optimal solution of $Q(\mathbf{f},\mathbf{z})$ such that*

$$\sum_{j\in V} d_j = \bar{d} \tag{6.1}$$

Proof. It is immediate since adding demand on a node will either change nothing to the number of required splitters, or increase the total splitter cost. □

Even though it was to be expected, this proposition greatly restricts the number of useful solutions in $\mathfrak{D}$.

Definition 6.1.1. *Let $G^*_{\mathbf{f},\mathbf{z}} = (V^*_I \cup V^*_J, E^*)$ be the bipartite graph derived from the initial graph $G = (V^* \cup \{0\}, E)$ in which the robust problem is solved. Set:*

- *$V^*_I = \{v \in V \ s.t. \ \tilde{\boldsymbol{z}} \neq 0\}$, the splitters locations. The maximum number of splitters that can be located in each node is given by the vector $\tilde{\mathbf{z}}$.*
- *$V^*_J = \{v \in V \ s.t. \ d^{max} \neq 0\}$, the possible demands locations.*
- *e^* is an edge of E^* if and only if $\tilde{\mathbf{f}} \neq 0$. The capacity of each edge is given by the vector $\tilde{\mathbf{f}}$. We denote by N_G the number of connected components, and H_p for $p = 1, ..., N_G$ the connected components of $G^*_{\mathbf{f},\mathbf{z}}$. We also need to introduce $V^{p*}_I = V^*_I \cap H_p$ and $V^{p*}_J = V^*_J \cap H_p$.*

Figure 6.1 sums up the elements introduced in definition 6.1.1 in a short example.

Now that we determined that an optimal solution of $Q(\mathbf{f},\mathbf{z})$ always consists in $\bar{d}$ demands that are distributed among the nodes of V^*_J, and that the graph can be decomposed, let us show that the recourse problem itself can be decomposed as well.

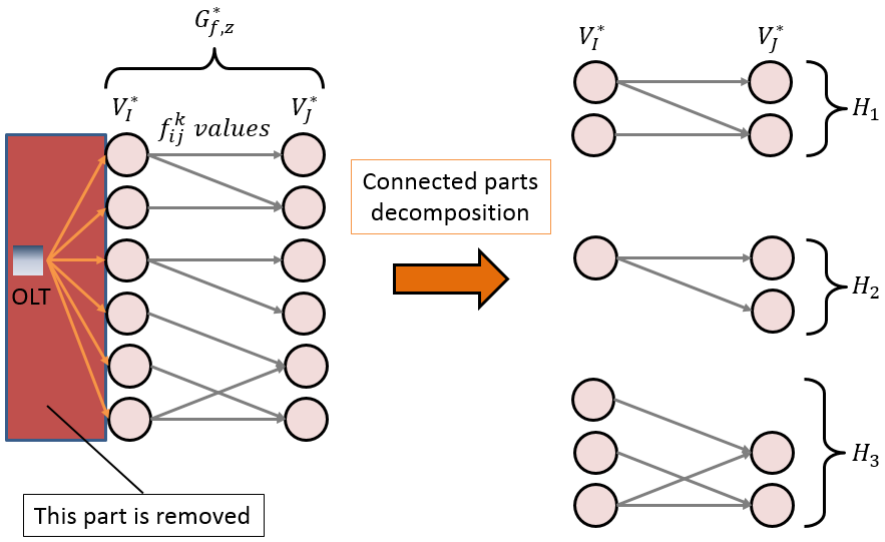

Figure 6.1: Example of $G^*_{\mathbf{f},\mathbf{z}}$ creation and decomposition into sub-connected-parts

Definition 6.1.2. *Let us denote by q_u^p the minimal number of demands to be distributed among the demand nodes of V_J^{p*} so that the minimal number of optical splitters needed to supply it equals u. We denote by $\boldsymbol{d}_u^p$ the demand vector on H_p that verifies this property, that is we have $\sum_{j \in V_J^*} \boldsymbol{d}_{j,u}^p = q_u^p$.*

In other words, q_u^p values are the optimal solution of the following bi-level problem denoted (Pq_u^p):

$$(Pq_u^p) \begin{cases} \min\limits_{d} & \sum\limits_{j \in V_J^{p*}} \boldsymbol{d}_j \\ \text{s.t.} & \boldsymbol{d} \in \mathfrak{D} \\ & \min\limits_{(\boldsymbol{\varphi}, \boldsymbol{\zeta}) \in P_{H_p}(\boldsymbol{d})} \left(\sum\limits_{i \in V_I^{p*}} \boldsymbol{\zeta}_i \right) \geq u \\ & \boldsymbol{\zeta} \leq \mathbf{z},\ \boldsymbol{\varphi} \leq \mathbf{f} \end{cases} \tag{6.2}$$

Constraints (6.2) ensures that $\boldsymbol{d}$ corresponds to a demand vector over which splitter location has been optimized. Otherwise, sub-optimal splitter location could be used in order to install too much splitters for a given demand vector.

For now, let us not worry about how to solve (Pq_u^p), and let us assume that q_u^p values are available to us.

Definition 6.1.3. *We denote by M_{H_p} a bound on the maximal number of splitters that can be needed in H_p for any demand scenario. Note that $M_{H_p} = \sum_{i \in V_I^{p*}} \mathbf{z}_i$ is a possible value.*

The following proposition holds:

Proposition 6.1.2. *The optimal solution's value of $Q(\mathbf{f}, \mathbf{z})$ equals the optimal solution's value of problem $Q_{decomp}(\mathbf{f}, \mathbf{z})$:*

$$Q_{decomp}(\mathbf{f}, \mathbf{z}) \begin{cases} \max\limits_{\mathbf{x}} & \sum\limits_{p=1}^{N_G} \sum\limits_{u=0}^{M_{H_p}} C \mathbf{x}_u^p u \\ s.t. & \sum\limits_{u=0}^{M_{H_p}} \mathbf{x}_u^p = 1, \ \forall p = 1, ..., N_G \quad (6.3) \\ & \sum\limits_{p=1}^{N_G} \sum\limits_{u=0}^{M_{H_p}} \mathbf{x}_u^p q_u^p \leq \bar{d} \quad (6.4) \\ & \mathbf{x}_u^p \in \{0, 1\} \end{cases}$$

And the optimal solution of $Q(\mathbf{f}, \mathbf{z})$ is the combination of $\boldsymbol{d}_u^p$ vectors associated to each $\mathbf{x}_u^p$ that equals 1.

Proof. For every H_p, q_u^p values gives us the best way to use a minimal amount of demand for all splitter numbers possible. Moreover, as H_p are not connected to each other, the demand scattering over V_J^{p*} nodes does not impact the best splitter solutions of the other connected parts of the graph. Of course, among all the possible splitter numbers to install on a given H_p, only one shall be chosen. We denote by $\mathbf{x}_u^p$ the boolean variable that equals one if the demand vector $\boldsymbol{d}_u^p$ is used for the solution in the sub-graph H_p. Constraint (6.3) ensures that only one variable $\mathbf{x}_u^p$ is set to one. Constraint (6.4) ensures that the total demand that is used in all H_p sub-graphs does not exceed the maximum demand possible $\bar{d}$. M_{H_p} gives a bound of the maximum number of splitters that can be used in a sub-graph H_p. □

In other words, if one is able to find q_u^p values for all u and p, then he would only have to solve $Q_{decomp}(\mathbf{f}, \mathbf{z})$ in order to solve $Q(\mathbf{f}, \mathbf{z})$. $Q_{decomp}(\mathbf{f}, \mathbf{z})$ is a kind of Knapsack problem with an additional constraint (6.3) that limits the total number of objects one can pick. It is a NP-hard problem, however, we expect it to be very tractable in practice, especially when N_G is low. Of course, the difficulty of the recourse problem has been moved elsewhere. Indeed, the real hard part now is to find q_u^p values. However, our goal is to design a heuristic for this problem. Therefore, we will now try to find "good" q_u^p values, found through a heuristic process. The reason for such problem decomposition shall be clearer for the reader in the following as it makes heuristic reasoning much easier now.

Let us denote by $\hat{q}_u^p$ the heuristic values we aim to find, and by $\hat{Q}_{decomp}(\mathbf{f}, \mathbf{z})$ the problem $Q_{decomp}(\mathbf{f}, \mathbf{z})$ when $\hat{q}_u^p$ values are used instead of q_u^p. The corresponding demand vector associated to a $\hat{q}_u^p$ value is denoted by $\hat{\boldsymbol{d}}_u^p$.

Proposition 6.1.3. *If $\hat{q}_u^p \geq q_u^p$ for all $p = 1, ..., N_G$ and for all $u = 1, ..., M_{H_p}$, then $\hat{Q}_{decomp}(\mathbf{f}, \mathbf{z})$ gives a lower bound of the recourse problem $Q(\mathbf{f}, \mathbf{z})$.*

Proof. If $\hat{q}_u^p \geq q_u^p$, $\forall p = 1, ..., N_G, \forall u = 1, ..., M_{H_p}$ then $\hat{Q}_{decomp}(\mathbf{f}, \mathbf{z})$ is more constrained than $Q_{decomp}(\mathbf{f}, \mathbf{z})$, whose optimal solution's value equals that of $Q(\mathbf{f}, \mathbf{z})$ (by Proposition 6.1.2). Therefore, it is a lower bound for that problem. □

From now on, our purpose will be to improve as much as possible our sequence of $\hat{q}_u^p$ values, using properties of the graph.

6.1.2 Improvement of the sequence by solving the maximal stable problem in a hypergraph

Proposition 6.1.4. *The following values for $\hat{q}_u^p$ are all greater or equal than q_u^p values:*

$$\hat{q}_0^p = 0 \tag{6.5}$$

$$\hat{q}_1^p = 1 \tag{6.6}$$

$$\hat{q}_u^p = 1 + (u-1) \times m, \forall u \geq 2 \tag{6.7}$$

This is true for all sub graphs H_p. Moreover, if there exists a demand vector $\hat{\boldsymbol{d}}_u^p \in \mathfrak{D}$ that verifies those equalities for a given u value, it can be associated to these heuristic values.

Proof. For $u = 0$, it is actually obvious that the minimal demand needed here also equals 0. In the same way, putting 1 demand anywhere in V_J^{p*} will directly imply the installation of 1 optical splitter in V_I^{p*}. More generally, putting $1 + (u-1) \times m$ demands in V_J^{p*}, even randomly distributed, will imply the use of at least u optical splitters to supply the necessary capacity. □

Remark 6.1.1. *Note that $q_0^p = \hat{q}_0^p = 0$ and $q_1^p = \hat{q}_1^p = 1$.*

Remark 6.1.2. *Note that $\hat{\boldsymbol{d}}_u^p$ can also be chosen so that $0 \leq \hat{\boldsymbol{d}}_{j_u}^{\ p} \leq d_j^{max}$, $\forall j \in V_J^{p*}$, as solving problem $Q_{decomp}(\mathbf{f}, \mathbf{z})$ will ensure that the chosen combination of $\hat{\boldsymbol{d}}_u^p$ will result in a demand vector in $\mathfrak{D}$.*

Proposition 6.1.4 gives us an easy-to-build sequence of heuristic $\hat{q}_u^p$ values to which any random $\hat{\boldsymbol{d}}_u^p$ can be associated. Thus, it will be the first step of our heuristic. Once it is done, we will improve the sequence locally. Let us start by the first values of the sequence, for low values of u, as the principle used to pose $\hat{q}_1^p = 1$ can be generalized.

Definition 6.1.4. *Let us denote by $\mathfrak{H}_p(V_J^{p*}, E^p)$ the hypergraph associated to H_p that is built by creating one hyperedge e_i^p per node i in V_I^{p*} and that contains the nodes of V_J^{p*} such that $e_i^p = \{j \in V_J^{p*} \,|\mathbf{f}_{ij} \geq 1\}$. To each hyperedge e_i^p is associated a capacity $F_i = \sum_{j \in V_J^{p*}} \min\{\mathbf{f}_{ij}, d_j^{max}\}$.*

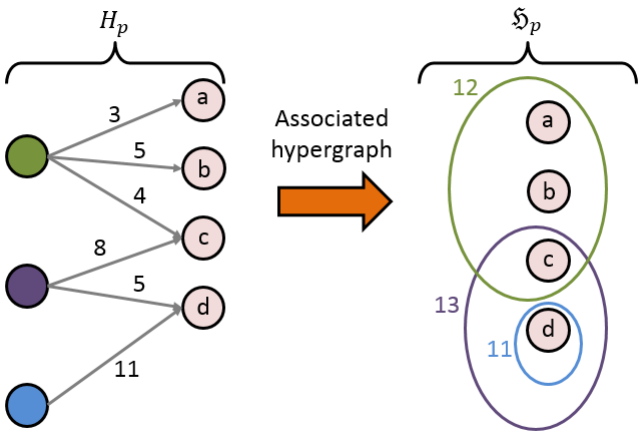

Figure 6.2: Illustration of how the associated hypergraph $\mathfrak{H}_p(V_J^{p*}, E^p)$ is built from H_p

Figure 6.2 gives an example of how that hypergraph is built. On the example showed by the figure, one can easily see that $q_2^p = 2$. Indeed, by putting 1 demand at node a or b, and 1 at node d, 2 optical splitters will be needed to supply that demand vector. This structure can be seen as a generalization of the stable of a graph to the hypergraph. Many definitions of the stable problem in a hypergraph are possible, and the reader may refer to the book of Claude Berge [8] for an extensive review. Here, let us define the stable problem in an hypergraph as follows:

Definition 6.1.5. *A stable S_p in a hypergraph $\mathfrak{H}_p(V_J^{p*}, E^p)$ is a subset of V_J^{p*} such that there is no hyperedge of E^p that can contain two elements of S_p. A stable of maximum cardinality is denoted by $\bar{S}_p$ and the problem that consists in finding one is denoted by (PS_p).*

Given the hypergraph $\mathfrak{H}_p(V_J^{p*}, E^p)$, let us denote by $\mathbf{s}_j$ the boolean variable that equals 1 if, and only if the node $j \in V_J^{p*}$ belongs to the maximum stable. Thus, the problem (PS_p) can be modeled as such:

$$(PS_p) \begin{cases} \max\limits_{\mathbf{s}} & \sum\limits_{j \in V_J^{p*}} \mathbf{s}_j \\ \text{s.t.} & \sum\limits_{j \in e_i^p} \mathbf{s}_j \leq 1, \ \forall e_i^p \in E^p \\ & \mathbf{s}_j \in \{0,1\} \end{cases}$$

(PS_p) is a NP-hard problem but in practice, we expect to get small hypergraphs $\mathfrak{H}_p(V_J^{p*}, E^p)$, thus making this procedure tractable. Moreover, stable like constraints are often well managed by up-to-date integer linear programming solvers. The following proposition tells us how the solution of (PS_p) will be used to improve $\hat{q}^p$ values.

Proposition 6.1.5. *Given $\bar{S}_p$, the following values for $\hat{q}_u^p$ are all greater or equal than q_u^p values:*

$$\hat{q}_u^p = u, \qquad \forall u = 1, ..., |\bar{S}_p| \tag{6.8}$$

$$\hat{q}_u^p = |\bar{S}_p| + (u - |\bar{S}_p|) \times m, \quad \forall u = |\bar{S}_p| + 1, ..., \left\lceil \frac{\sum_{j \in \bar{S}_p} d_j^{max}}{m} \right\rceil \tag{6.9}$$

For $u = 1, ..., |\bar{S}_p|$, the associated demand vectors are those with 1 demand per nodes of the stable $\bar{S}_p$. For $u = |\bar{S}_p| + 1, ..., \left\lceil \frac{\sum_{j \in \bar{S}_p} d_j^{max}}{m} \right\rceil$, the associated demand vectors are those with 1 plus a multiple of m demands put on one node of the stable $\bar{S}_p$.

Proof. By construction, by putting 1 demand at each node of the stable $\bar{S}_p$, each demand will imply the installation of a splitter to supply it. That is because nodes of the stable can not be supplied by the same splitter node. Once all nodes of the stable received their single demand, adding m demands to a node of the stable will imply at least one more optical splitter to be installed in order to supply it. By fulfilling nodes of $\bar{S}_p$ up to their maximal demand, we can keep increasing the number of splitters required to supply them as well. □

Looking at the figure 6.2, the maximum stable here would be, for example, the set of nodes composed of nodes a and d. By putting 1 demand on a and d, the only mean for supplying them is to put one splitter on the green node and one on the purple (or blue) node. Then, by adding m demands on a or d, another optical splitter will be needed to supply it. We can keep applying this rule in order to find lower values for $\hat{q}_u^p$ than those found in Proposition 6.1.4, for all $u = 1, ..., \left\lceil \frac{\sum_{j \in \bar{S}_p} d_j^{max}}{m} \right\rceil$. If the stable is of size 1, then proposition 6.1.5 has no impact on $\hat{q}_u^p$ values.

Remark 6.1.3. *There may be several maximum stables for a given hypergraph $\mathfrak{H}_p$. If that happens, one should pick the one that has the higher $\left\lceil \frac{\sum_{j \in \bar{S}_p} d_j^{max}}{m} \right\rceil$ value. This can be done by a goal programming approach. One just need to solve (PS_p) once, and then again by changing the objective function as such:*

$$\max_{\mathbf{s}} \sum_{j \in V_J^{p*}} d_j^{max} \mathbf{s}_j$$

and adding the following constraint to the problem:

$$\sum_{j \in V_J^{p*}} \mathbf{s}_j = |\bar{S}_p|$$

Remark 6.1.4. *For all $u = 1, ..., |\bar{S}_p|$, $\hat{q}_u^p$ values given by Proposition 6.1.5 are actually optimal and equal to q_u^p. Indeed, spending 1 demand to cause 1 more splitter is the best ratio one could have here.*

6.1.3 Improvement of the sequence by exploiting F_i values

Another way for improving the sequence of $\hat{q}_u^p$ values is to make use of the fiber capacities. Indeed, $\mathbf{f}_{ij}$ values impose some limitations on the routing options for minimizing the splitter installation cost. Thus, there may exist some demand affectations that will restrict the number of routing options so that the splitter cost minimization problem will have to use more splitters than expected. In order to have a fast way of making use of these capacity limitations, we need generic procedure. For each hyperedge, to which is associated a capacity F_i, we denote by F_i^r the remainder of the division of F_i by m. Assume, without loss of generality, that F_i^r values are sorted, such that $F_{(i)}^r \geq F_{(i+1)}^r$. We thus have the following proposition:

Proposition 6.1.6. *Given $\tilde{u}^p = \sum_{i \in V_I^{p*}} \left\lfloor \frac{F_i}{m} \right\rfloor$, if $\tilde{u}^p < M_{H_p}$ then the following values for $\hat{q}_u^p$ are all greater or equal than q_u^p values:*

$$\hat{q}^p_{\tilde{u}^p+1} = 1 + \tilde{u}^p \times m \tag{6.10}$$

$$\hat{q}^p_{\tilde{u}^p+i} = \hat{q}^p_{\tilde{u}^p+i-1} + F^r_{(i-1)},\ \forall i = 2, ..., M_{H_p} - (\tilde{u}^p + 1) \tag{6.11}$$

This is true for all sub graphs H_p. Moreover, any demand vector $\hat{\boldsymbol{d}}_u^p \in \mathfrak{D}$ that verifies those equalities can be associated to these heuristic values.

Proof. Once $1 + \tilde{u}^p \times m$ demands are placed on the nodes of V_J^{p*}, in the best case they are distributed so that every hyperedge e_i^p contains exactly $\left\lfloor \frac{F_i}{m} \right\rfloor \times m$ demands. Therefore, only adding the greatest reminder $F_{(1)}^r$ to any node of the hyperedge (1) will add at least one splitter because this hyperedge will be saturated. We can iterate the process until the maximal u value is reached. As this is true for the specific demand vector that allocate $\left\lfloor \frac{F_i}{m} \right\rfloor \times m$ demands to each hyperedge, and then $1 + F_{(1)}^r$ to the edge (1), and then $F_{(2)}^r$ to the edge (2), and so on, which is the worst case for us, it is also true for any random demand vector in $\mathfrak{D}$ since it can only be better. □

Figure 6.3 gives an example of how this principle is applied. At first, 1 demand is allocated to node a. Then, $\left\lfloor \frac{10}{8} \right\rfloor \times 8 = 8$ demands are allocated to node a. At this point, only 4 demands are needed to saturate the hyperedge c. Therefore, adding 4 demands (instead of 8) is enough to saturate the edge and imply another splitter somewhere on

node d or e. Finally, 2 demands are necessary on node b to saturate another hyperedge (that is either d or e). Adding more demand would prove irrelevant since they would not imply any more splitter.

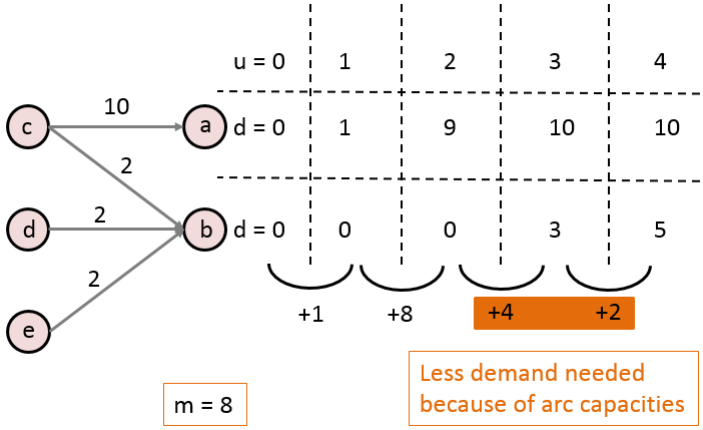

Figure 6.3: Example of how the Proposition 6.1.6 may be used to reduce the $\hat{q}_u^p$ values

Note that the demand allocation proposed in Figure 6.3 is not optimal. Figure 6.4 shows an optimal $\hat{q}_u^p$ sequence that uses less demand.

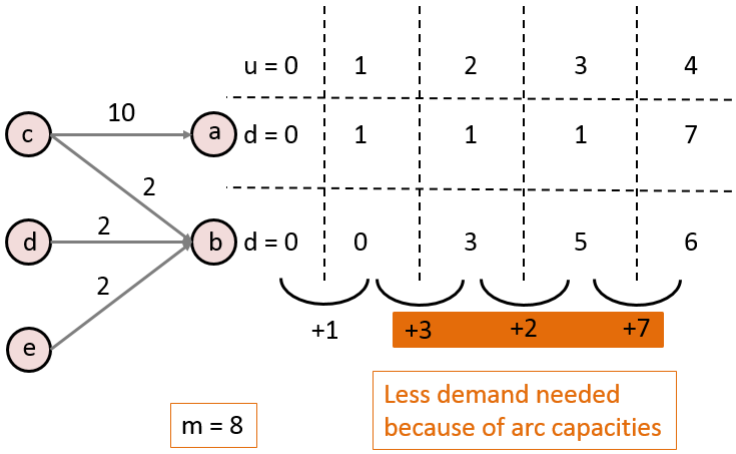

Figure 6.4: Optimal sequence of $\hat{q}_u^p$ values for the example shown in Figure 6.3

6.1.4 Design of the heuristic for the recourse problem

Many principles of the heuristic were introduced in the last subsections. Let us sum them up in this section, as they are presented in Algorithm 10.

Experimental testing of the heuristic

In order to estimate how good the heuristic performs, we needed to compare it to the optimal solutions of the recourse problem. However, as these solutions are hard to obtain on graphs larger than 10 nodes, the heuristic could be tested only on small instances. Therefore, the following results have to be considered regarding that fact. Indeed, since worst case scenarios imply a low number of splitters for theses instances, the relative gap between the heuristic and the optimal solution can increase very fast for each worst case splitter that is not grasped by the heuristic. Thus, we estimated it more relevant to report the percentage of cases where the heuristic found the right number of splitters in the worst case scenario.

To that extent, we launched algorithm 6 over 50 randomly generated instances (between 5 and 10 nodes) and, at each iteration of the master problem, we launched solved the recourse problem both exactly and with the heuristic. Thus, they were compared over (around) 1000 cases. We observed that most of the time (97%), demand scenarios that were generated were different. However, the worst case cost found by the heuristic equaled the optimal one in 81% of cases. Most cases where the heuristic performs well are in the first iterations of the main algorithm since graphs have a lot of non connected parts H_p with only one demand node, thus making the heuristic optimal for those sub-parts. As the algorithm goes on, stables are detected by the heuristic that keep on performing quite well. But after a while, as the solution built by the algorithm gets more and more "robust", the heuristic struggles to find good q_u^p values.

Let us now look at the solving time for the heuristic. Note that it highly depends on the structure of the graph we are solving. Indeed, finding the maximum stable is a NP-hard problem and it may prove hard to solve for large H_p components. However, we experimentally witnessed the fact that the size of H_p components does not increase a lot, except for one component that tends to contain a lot more nodes than the others (that remain with a size below 5 nodes in general). However, solving $\hat{Q}_{decomp}(\mathbf{f}, \mathbf{z})$ proves harder as the graph increases in size. Figure 6.5 shows how the solving times increases with the instance size. The solving time showed on the figure is the average solving time over 10 randomly generated instances per given size. The first thing to note is the exponential increase of the solving time with the size of the graph. However, for the higher graph size we tested here, that is graphs of 600 nodes, the solving time remained below 35 seconds on average.

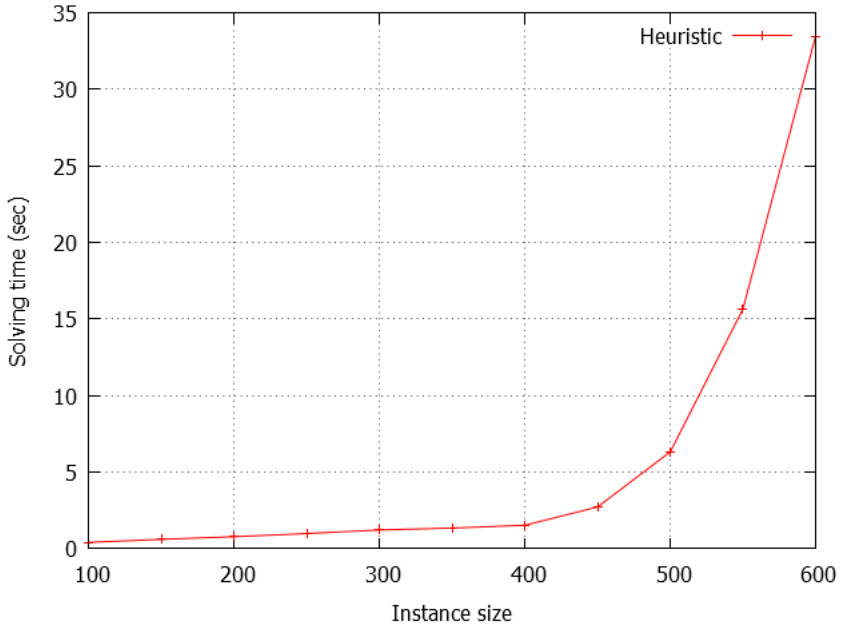

Figure 6.5: Average solving time of the heuristic (Algorithm 10)

6.2 Non-exact approaches based on relaxations of the recourse problem

In this section, we explore various relaxations and approaches for tackling the recourse problem. Methods will be presented and at the end of the section, compared so that the same experimental protocol is applied to all.

6.2.1 Using the Algorithm 10 for solving the recourse problem within Algorithm 6

Algorithm 10 designed in the last section can be used in many ways. For example, it could be used to increase its speed by generating the first scenarios by the means of the heuristic instead of the heavy exact method. But as we aim for non exact solving approaches, we will use the heuristic in the initially exact framework developed in Chapter 5 to solve the robust problem approximately. Indeed, the heuristic can be used to generate demand scenarios and adding them to (PON_{rob}) (see Section 5.1). Of course, as the heuristic only provides a lower bound for the recourse problem, it is not possible to prove the convergence of Algorithm 6 in that case. Therefore, a stopping criterion must be used in order for the algorithm to stop. Here, we shall simply limit the number of iteration that the framework will operate before stopping at a given number L. That modified,

heuristic version of Algorithm 6 is given by Algorithm 11.

6.2.2 Continuous relaxation of the recourse problem and dualization

What makes the robust problem (PON_{rob}) hard to solve is, among many things, the integrity of some recourse variables. This prevented us from using the algorithms available in the literature (see Section 1.2.6). Thus, by relaxing the integrity constraint of recourse variables $\boldsymbol{\zeta}$, it would be possible to use an adaptation of the generic Algorithm 1 presented in Section 1.2.6. Let us consider the recourse problem of (PON_{rob}):

$$Q(\mathbf{f}, \mathbf{z}) \ : \max_{d \in \mathfrak{D}} \min_{\substack{(\boldsymbol{\varphi}, \boldsymbol{\zeta}) \in P_G(d) \\ \boldsymbol{\zeta} \leq \mathbf{z},\ \boldsymbol{\varphi} \leq \mathbf{f}}} \sum_{i \in V^*} C\boldsymbol{\zeta}_i$$

Proposition 6.2.1. *If recourse variables of the recourse problem have their integrity relaxed, Q can thus be written as such:*

$$Q_{relaxed}(\mathbf{f}, \mathbf{z}) \begin{cases} \max\limits_{\boldsymbol{d}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{\theta}} & \sum\limits_{j \in V^*} \boldsymbol{\mu}_j \boldsymbol{d}_j - \sum\limits_{i \in V^*} \boldsymbol{\theta}_i \mathbf{z}_i - \sum\limits_{(i,j) \in V^{*2}} \boldsymbol{\pi}_{ij} \mathbf{f}_{ij} & \\ s.t. & m\boldsymbol{\lambda}_i - \boldsymbol{\theta}_i \leq C,\ \forall i \in V^* & (6.12) \\ & \boldsymbol{\mu}_j \leq \boldsymbol{\lambda}_i + \boldsymbol{\pi}_{ij},\ \forall (i,j) \in V^{*2} & (6.13) \\ & 0 \leq \boldsymbol{d}_j \leq d_j^{max},\ \forall j \in V^* & (6.14) \\ & \sum\limits_{j \in V^*} \boldsymbol{d}_j \leq \bar{d} & (6.15) \\ & \boldsymbol{d}_j \in \mathbb{N},\ \forall j \in V^* & \\ & \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{\theta} \geq 0 & \end{cases}$$

Proof. Let us write the inner relaxed minimization problem:

$$\begin{aligned} \min_{\boldsymbol{\varphi}, \boldsymbol{\zeta}} & \sum_{i \in V^*} C\boldsymbol{\zeta}_i & \\ \text{s.t.} & \sum_{j \in V^*} \boldsymbol{\varphi}_{ij} \leq m\boldsymbol{\zeta}_i,\ \forall i \in V^* & \leftarrow \lambda_i \\ & \sum_{i \in V^*} \boldsymbol{\varphi}_{ij} \geq \boldsymbol{d}_j,\ \forall j \in V^* & \leftarrow \mu_j \\ & 0 \leq \boldsymbol{\varphi}_{ij} \leq \mathbf{f}_{ij},\ \forall (i,j) \in V^{*2} & \leftarrow \pi_{ij} \\ & 0 \leq \boldsymbol{\zeta}_i \leq \mathbf{z}_i,\ \forall i \in V^* & \leftarrow \theta_i \end{aligned}$$

By strong duality, this problem's optimal value equals its dual. Dual variables are shown on the right of each constraint. □

The objective function of $Q_{relaxed}(\mathbf{f}, \mathbf{z})$ contains quadratic terms. Such situation has already been encountered in Section 5.2.2. Thus, we do not enter in the details of the linearization again. The reader may just remember that the quadratic terms can be linearized by decomposing each variable $\boldsymbol{d}_j$ into its binary writing, as for the $(\hat{P}_{mult})$ formulation shown in Section 5.2.2. In that case, it is possible to use a column-and-constraint generation algorithm for solving that relaxed version of (PON_{rob}). Algorithm 12, presented above, is precisely doing this. It is directly derived from Algorithm 1 of Section 1.2.6, which is why we skip the details of the procedure.

6.2.3 Affine approximation on the recourse variables

For now, every methods we proposed for solving the two-stage robust problem are based on column-and-constraint generation. However, for a non-exact approach, there are other ways proposed in the literature, such as the affine approximation on the recourse variables. Indeed, as it is presented in the bibliography section 1.2.6, by making the assumption that recourse variables follow a certain policy, it is possible to rewrite the problem as a smaller, hopefully tractable one. Many policies are possible, but the one that has been chosen is the affine approximation policy, mostly for practical reasons since it enables one to keep a linear programming approach in order to solve it.

Let us recall that in (PON_{rob}), there are two main sets of second stage variables:

- $\boldsymbol{\varphi}_{ij}(\boldsymbol{d})$ the number of fibers routed along the shortest path from splitters at node i to the demand node j for the demand scenario $\boldsymbol{d}$
- $\boldsymbol{\zeta}_i(\boldsymbol{d})$ the number of splitters eventually installed at node i in order to supply the demand scenario $\boldsymbol{d}$

Here, recourse variables are written as functions of the uncertain vector $\boldsymbol{d}$, which just means that they may take different values for each scenario. The affine approximation on recourse variables consists in stating that recourse variables actually are an affine function of the uncertainty. That is, recourse variables are defined as in the following hypothesis:

Hypothesis 4. *Let us suppose that recourse variables of the problem (PON_{rob}) are affine functions of the uncertainty. They are written as follows:*

$$\boldsymbol{\varphi}_{ij}(\boldsymbol{d}) = \boldsymbol{\varphi}^0_{ij} + \sum_{h \in V^*} \boldsymbol{\varphi}^h_{ij} \boldsymbol{d}_h, \ \forall (i,j) \in V^{*2} \tag{6.16}$$

$$\boldsymbol{\zeta}_i(\boldsymbol{d}) = \boldsymbol{\zeta}^0_i + \sum_{h \in V^*} \boldsymbol{\zeta}^h_i \boldsymbol{d}_h, \ \forall i \in V^* \tag{6.17}$$

with all $\boldsymbol{\varphi}^h_{ij}$ and $\boldsymbol{\zeta}^h_i$ having no sign for all $h = 0, ..., |V^*|$.

Remark 6.2.1. *Note that imposing $\boldsymbol{\zeta}_i(\boldsymbol{d}) \in \mathbb{N}$ for all $\boldsymbol{d} \in \mathfrak{D}$ would imply that $\boldsymbol{\zeta}_i^h = 0$ for all $h \in V^*$. Hence, the interest of the affine approximation is lost in the process. Therefore, the integrity of recourse variables must be considered relaxed here.*

Remark 6.2.1 could make one feel that the affinely adjustable robust approach implies too much approximation to be relevant. Since in this section, we try to find tractable approaches for "good" solutions, at least trying the affinely robust one is relevant, the only criteria being the experimental results that will follow. However, this approach is also relevant because it has the potential to perform best, both computationally and regarding the result since, for example, one can easily show that the optimal solution of the affine version of (PON_{rob}) will always be greater or equal than the integrity-relaxed recourse version of it (presented in the last section 6.2.2). However, it may be either an upper bound or a lower bound for the problem. Here again, experimental results will separate the wheat from the chaff.

Under the hypothesis 4, (PON_{rob}) become another problem, denoted (PON_{AARC}) for Affinely Adjustable Robust Counterpart, and can be written as follows:

$$
(PON_{AARC}) \begin{cases}
\min\limits_{\mathbf{z},\mathbf{f},\boldsymbol{\zeta},\boldsymbol{\varphi}} & \sum\limits_{(i,j)\in V^{*2}} c_{ij}\mathbf{f}_{ij} + \sum\limits_{i\in V^*} c_{0i}\mathbf{z}_i + \boldsymbol{\gamma} & \\
\text{s.t.} & \boldsymbol{\gamma} \geq C \sum\limits_{i\in V^*} \left(\boldsymbol{\zeta}_i^0 + \sum\limits_{h\in V^*} \boldsymbol{\zeta}_i^h \boldsymbol{d}_h \right),\ \forall \boldsymbol{d} \in \mathfrak{D} & (6.18)\\
& \sum\limits_{j\in V^*} \left(\boldsymbol{\varphi}_{ij}^0 + \sum\limits_{h\in V^*} \boldsymbol{\varphi}_{ij}^h \boldsymbol{d}_h \right) \leq m \left(\boldsymbol{\zeta}_i^0 + \sum\limits_{h\in V^*} \boldsymbol{\zeta}_i^h \boldsymbol{d}_h \right),\ \forall i \in V^*, \forall \boldsymbol{d} \in \mathfrak{D} & (6.19)\\
& \sum\limits_{i\in V^*} \left(\boldsymbol{\varphi}_{ij}^0 + \sum\limits_{h\in V^*} \boldsymbol{\varphi}_{ij}^h \boldsymbol{d}_h \right) \geq \boldsymbol{d}_j,\ \forall j \in V^*, \forall \boldsymbol{d} \in \mathfrak{D} & (6.20)\\
& \boldsymbol{\varphi}_{ij}^0 + \sum\limits_{h\in V^*} \boldsymbol{\varphi}_{ij}^h \boldsymbol{d}_h \leq \mathbf{f}_{ij},\ \forall (i,j) \in V^{*2}, \forall \boldsymbol{d} \in \mathfrak{D} & (6.21)\\
& \boldsymbol{\zeta}_i^0 + \sum\limits_{h\in V^*} \boldsymbol{\zeta}_i^h \boldsymbol{d}_h \leq \mathbf{z}_i,\ \forall i \in V^*, \forall \boldsymbol{d} \in \mathfrak{D} & (6.22)\\
& \boldsymbol{\varphi}_{ij}^0 + \sum\limits_{h\in V^*} \boldsymbol{\varphi}_{ij}^h \boldsymbol{d}_h \geq 0,\ \forall (i,j) \in V^{*2}, \forall \boldsymbol{d} \in \mathfrak{D} & (6.23)\\
& \boldsymbol{\zeta}_i^0 + \sum\limits_{h\in V^*} \boldsymbol{\zeta}_i^h \boldsymbol{d}_h \geq 0,\ \forall i \in V^*, \forall \boldsymbol{d} \in \mathfrak{D} & (6.24)\\
& (\mathbf{f}, \mathbf{z}) \in P_G(d^{max}) & \\
& \mathbf{z} \in \mathbb{N}^{|V^*|},\ \mathbf{f} \in \mathbb{R}_+^{|V^{*2}|} &
\end{cases}
$$

As such, (PON_{AARC}) is not a linear problem. The following Theorem shows how to transform it so it can be tackled by a classic MIP solving approach.

Theorem 6.2.1. *(PON_{AARC}) can be rewritten as follows:*

$$
(PON_{AARC})\begin{cases}
\min\limits_{\mathbf{z},\mathbf{f},\boldsymbol{\zeta},\boldsymbol{\varphi},\mathbf{p},\mathbf{t}} & \sum\limits_{(i,j)\in V^{*2}} c_{ij}\mathbf{f}_{ij} + \sum\limits_{i\in V^*} c_{0i}\mathbf{z}_i + \boldsymbol{\gamma} & \\
s.t. & \gamma \geq C\sum\limits_{i\in V^*}\boldsymbol{\zeta}_i^0 + \mathbf{t}^\gamma \bar{d} + \sum\limits_{h\in V^*}\mathbf{p}_h^\gamma d_h^{max} & (6.25)\\
& \mathbf{p}_h^\gamma + \mathbf{t}^\gamma \geq \sum\limits_{i\in V^*}\boldsymbol{\zeta}_i^h,\ \forall h\in V^* & (6.26)\\
& \sum\limits_{j\in V^*}\boldsymbol{\varphi}_{ij}^0 + \mathbf{t}_i^m\bar{d} + \sum\limits_{h\in V^*}\mathbf{p}_{h,i}^m d_h^{max} \leq m\boldsymbol{\zeta}_i^0,\ \forall i\in V^* & (6.27)\\
& \mathbf{p}_{h,i}^m + \mathbf{t}_i^m + \boldsymbol{\zeta}_i^h \geq \sum\limits_{j\in V^*}\boldsymbol{\varphi}_{ij}^h,\ \forall i\in V^*, \forall h\in V^* & (6.28)\\
& \sum\limits_{i\in V^*}\boldsymbol{\varphi}_{ij}^0 \geq \sum\limits_{h\in V^*}\mathbf{p}_{h,j}^d d_h^{max} + \mathbf{t}_j^d\bar{d},\ \forall j\in V^* & (6.29)\\
& \mathbf{p}_{h,j}^d + \mathbf{t}_j^d + \sum\limits_{i\in V^*}\boldsymbol{\varphi}_{ij}^h \geq 0,\ \forall j\in V^*, \forall h\in V^*\backslash\{j\} & (6.30)\\
& \mathbf{p}_{h,j}^d + \mathbf{t}_j^d + \sum\limits_{i\in V^*}\boldsymbol{\varphi}_{ij}^h \geq 1,\ \forall j\in V^*, h=j & (6.31)\\
& \boldsymbol{\varphi}_{ij}^0 + \mathbf{t}_{ij}^f\bar{d} + \sum\limits_{h\in V^*}\mathbf{p}_{h,ij}^f d_h^{max} \leq \mathbf{f}_{ij},\ \forall (i,j)\in V^{*2} & (6.32)\\
& \mathbf{p}_{h,ij}^f + \mathbf{t}_{ij}^f \geq \boldsymbol{\varphi}_{ij}^h,\ \forall (i,j)\in V^{*2}, \forall h\in V^* & (6.33)\\
& \boldsymbol{\zeta}_i^0 + \mathbf{t}_i^z\bar{d} + \sum\limits_{h\in V^*}\mathbf{p}_{h,i}^z d_h^{max} \leq \mathbf{z}_i,\ \forall i\in V^* & (6.34)\\
& \mathbf{p}_{h,i}^z + \mathbf{t}_i^z \geq \boldsymbol{\zeta}_i^h,\ \forall i\in V^*, \forall h\in V^* & (6.35)\\
& \boldsymbol{\varphi}_{ij}^0 \geq \mathbf{t}_{ij}^{f0}\bar{d} + \sum\limits_{h\in V^*}\mathbf{p}_{h,ij}^{f0} d_h^{max},\ \forall (i,j)\in V^{*2} & (6.36)\\
& \mathbf{p}_{h,ij}^{f0} + \mathbf{t}_{ij}^{f0} + \boldsymbol{\varphi}_{ij}^h \geq 0,\ \forall (i,j)\in V^{*2}, \forall h\in V^* & (6.37)\\
& \boldsymbol{\zeta}_i^0 \geq \mathbf{t}_i^{z0}\bar{d} + \sum\limits_{h\in V^*}\mathbf{p}_{h,i}^{z0} d_h^{max},\ \forall i\in V^* & (6.38)\\
& \mathbf{p}_{h,i}^{z0} + \mathbf{t}_i^{z0} + \boldsymbol{\zeta}_i^h \geq 0,\ \forall i\in V^*, \forall h\in V^* & (6.39)\\
& (\mathbf{f},\mathbf{z}) \in P_G(d^{max}) & (6.40)\\
& \mathbf{z}\in\mathbb{N}^{|V^*|},\ \mathbf{f}\in\mathbb{R}_+^{|V^{*2}|} & \\
& \mathbf{p}\geq 0, \mathbf{t}\geq 0 &
\end{cases}
$$

Proof. The problem (PON_{AARC}) can first be rewritten as follows:

$$(PON_{AARC}) \begin{cases} \min\limits_{\mathbf{z},\mathbf{f},\boldsymbol{\zeta},\boldsymbol{\varphi}} & \sum\limits_{(i,j)\in V^{*2}} c_{ij}\mathbf{f}_{ij} + \sum\limits_{i\in V^*} c_{0i}\mathbf{z}_i + \boldsymbol{\gamma} \\ \text{s.t.} & \boldsymbol{\gamma} \geq C\sum\limits_{i\in V^*}\boldsymbol{\zeta}_i^0 + \max\limits_{\boldsymbol{d}\in\mathfrak{D}}\left(\sum\limits_{i\in V^*}\sum\limits_{h\in V^*}\boldsymbol{\zeta}_i^h\boldsymbol{d}_h\right) & (6.41) \\ & \sum\limits_{j\in V^*}\boldsymbol{\varphi}_{ij}^0 + \max\limits_{\boldsymbol{d}\in\mathfrak{D}}\left(\sum\limits_{h\in V^*}\left(-\boldsymbol{\zeta}_i^h + \sum\limits_{j\in V^*}\boldsymbol{\varphi}_{ij}^h\right)\boldsymbol{d}_h\right) \leq m\boldsymbol{\zeta}_i^0,\ \forall i\in V^* & (6.42) \\ & \sum\limits_{i\in V^*}\boldsymbol{\varphi}_{ij}^0 + \min\limits_{\boldsymbol{d}\in\mathfrak{D}}\left(\sum\limits_{i\in V^*}\sum\limits_{h\in V^*}\boldsymbol{\varphi}_{ij}^h\boldsymbol{d}_h\right) \geq \boldsymbol{d}_j,\ \forall j\in V^* & (6.43) \\ & \boldsymbol{\varphi}_{ij}^0 + \max\limits_{\boldsymbol{d}\in\mathfrak{D}}\left(\sum\limits_{h\in V^*}\boldsymbol{\varphi}_{ij}^h\boldsymbol{d}_h\right) \leq \mathbf{f}_{ij},\ \forall(i,j)\in V^{*2} & (6.44) \\ & \boldsymbol{\zeta}_i^0 + \max\limits_{\boldsymbol{d}\in\mathfrak{D}}\left(\sum\limits_{h\in V^*}\boldsymbol{\zeta}_i^h\boldsymbol{d}_h\right) \leq \mathbf{z}_i,\ \forall i\in V^* & (6.45) \\ & \boldsymbol{\varphi}_{ij}^0 + \min\limits_{\boldsymbol{d}\in\mathfrak{D}}\left(\sum\limits_{h\in V^*}\boldsymbol{\varphi}_{ij}^h\boldsymbol{d}_h\right) \geq 0,\ \forall(i,j)\in V^{*2} & (6.46) \\ & \boldsymbol{\zeta}_i^0 + \min\limits_{\boldsymbol{d}\in\mathfrak{D}}\left(\sum\limits_{h\in V^*}\boldsymbol{\zeta}_i^h\boldsymbol{d}_h\right) \geq 0,\ \forall i\in V^* & (6.47) \\ & (\mathbf{f},\mathbf{z}) \in P_G(d^{max}) \\ & \mathbf{z}\in\mathbb{N}^{|V^*|},\ \mathbf{f}\in\mathbb{R}_+^{|V^{*2}|} \end{cases}$$

Thus, in order to obtain a linear program, one needs to dualize every inner optimization problem of every constraints. Since they are all similar, let us give the main principle. All optimization problems here can be put into the following writing:

$$\max_{\boldsymbol{d}\in\mathfrak{D}}\sum_{h\in V^*} X_h d_h$$

whose dual thus is:

$$\begin{aligned} \min_{\mathbf{t},\mathbf{p}}\ & \mathbf{t}\bar{d} + \sum_{h\in V^*}\mathbf{p}_h d_h^{max} \\ \text{s.t.}\ & \mathbf{p}_h + \mathbf{t} \geq X_h \\ & \mathbf{p}_h \geq 0,\ \forall h\in V^* \\ & \mathbf{t}\geq 0 \end{aligned}$$

That last form can directly replace the inner optimization problems within the constraints. Of course, every optimization problem is different and a new set of dual variables $\mathbf{p}$ and $\mathbf{t}$ should be assigned to every problem. □

Remark 6.2.2. *The dual variables associated to inner optimization problems in constraints* (6.44) *and* (6.46) *(respectively* (6.45) *and* (6.47)*), are actually defining the same*

set. Therefore, $\mathbf{t}_{ij}^{f}$ *and* $\mathbf{p}_{h,ij}^{f}$ *(respectively* $\mathbf{t}_{i}^{z}$ *and* $\mathbf{p}_{h,i}^{z}$*) can be used instead of variables* $\mathbf{t}_{ij}^{f0}$ *and* $\mathbf{p}_{h,ij}^{f0}$ *(respectively* $\mathbf{t}_{i}^{z0}$ *and* $\mathbf{p}_{h,i}^{z0}$*), thus reducing the total number of variables.*

Remark 6.2.3. *The transformation proposed by Theorem 6.2.1 adds* $(2|V^*|^3 + 6|V^*|^2 + 4|V^*| + 1)$ *continuous variables and* $(2|V^*|^3 + 4|V^*|^2 + |V^*|)$ *constraints. With remark 6.2.2,* $(|V^*|^3 + 4|V^*|^2 + 4|V^*| + 1)$ *more continuous variables and* $(|V^*|^3 + 3|V^*|^2 + |V^*|)$ *constraints are needed.*

Thus, the problem is greatly increased in size, but it has the advantage to be solved only once (contrary to column-and-constraint generation approaches).

6.2.4 Experimental testing of relaxed approaches

Many approaches were designed in order to tackle (PON_{rob}), exactly or not. In order to compare them, one needs to design an experimental protocol that could expose strengths and weaknesses of all considered methods. These methods shall be compared over the following criteria:

- The quality of the solution, both in terms of cost and robustness
- The solving time
- The instance's size that may be tackled

As we are dealing with a two-stage problem, we have two kinds of costs: here-and-now fiber cost and future robust cost. Given a first stage solution, it is easy to give the first cost, but having the corresponding robust cost is difficult. Therefore, we need to design a method in order to estimate that robust cost, even approximately, but on a fair basis. Before entering the details, let us define our notations.

- The trivial method that consists in solving the problem without considering uncertainty, used here as the scientific control, is denoted by $PON_{trivial}$
- The method presented in Section 6.2.1, using the heuristic Algorithm 10, is denoted by PON_{heur}
- The method presented in Section 6.2.2, based on the continuous relaxation of the recourse problem, is denoted by PON_{relax}
- The method presented in Section 6.2.3, based on the affine approximation of recourse variables, is denoted by PON_{AARC}

Here, the method $PON_{trivial}$ consists in solving the following mixed integer linear program:

$$\min_{(\mathbf{f},\mathbf{z})\in P_G(d^{max})} \sum_{(i,j)\in V^{*2}} c_{ij}\mathbf{f}_{ij} + \sum_{i\in V^*}(c_{0i}+C)\mathbf{z}_i$$

and then extract the optimal solution $(\mathbf{f}^*, \mathbf{z}^*)$ as the first stage solution for (PON_{rob}). This way, this solution will give us the threshold that any approached robust method should not exceed. Indeed, if any of the above quoted methods would prove less effective than $PON_{trivial}$, which does not deal at all with demand uncertainty, then it would be enough to reject it.

Defining ways of estimating the robust cost

As mentioned before, obtaining the exact value of the robust problem for given values of $(\mathbf{f}, \mathbf{z})$ is hard. Therefore, there is a need for methods that can estimate that worst cost. One is already available to us: the Algorithm 10 designed in Section 6.1. Let us denote by v_r^h the value of the robust cost found by the algorithm 10. However, this is not enough, especially since method PON_{heur} makes an explicit use of it. Moreover, the heuristic is based on few causes of splitter cost increasing, but not all of them. Thus, in order to have two different estimations of the robust cost, let us propose a robust cost estimating method, based on the computation of the best splitter cost over a large sample of demand scenarios.

Indeed, solving the inner minimization problem of the recourse problem, that is:

$$\mathcal{R}(\mathbf{f},\mathbf{z},\boldsymbol{d}) = \left(\min_{\substack{(\boldsymbol{\varphi},\boldsymbol{\zeta})\in P_G(\hat{\boldsymbol{d}}) \\ \boldsymbol{\zeta}\leq\mathbf{z},\ \boldsymbol{\varphi}\leq\mathbf{f}}} \sum_{i\in V^*} C\boldsymbol{\zeta}_i \right)$$

is fast enough to make its solving over a large number of scenarios possible.

Let us denote by $\tilde{\mathfrak{D}}$ the subset of scenarios over which $\mathcal{R}(\mathbf{f},\mathbf{z},\boldsymbol{d})$ will be solved. Let us denote by v_r^s the value of the worst splitter cost found by solving $\mathcal{R}(\mathbf{f},\mathbf{z},\boldsymbol{d})$ over the sample of scenarios $\tilde{\mathfrak{D}}$. Moreover, having the average splitter cost is also interesting, since some methods may give solutions with a large worst splitter cost, but a good average splitter cost. Let us denote by $\bar{v}_r^s$ that mean cost.

Demand scenarios of $\tilde{\mathfrak{D}}$ are generated by algorithm 13.

Other random vector generating methods could have been chosen, but as no probability assumptions are made on the uncertain demand, we preferred to introduce as less bias as possible in the random generation.

Table 6.1 reports the comparison of all envisaged approaches of Section 6.2 to the optimal value found with Algorithm 6. For each tested instance (that is randomly generated), we give its size. Then, for each solving approach, we report its solving time in seconds (columns "Time") and the value it found (columns "v"). In the columns of (PON_{rob}) are reported the results found with Algorithm 6, in columns of "AARC" are reported the results found by solving (PON_{AARC}), in columns of "Heuristic" are reported the results found by using the heuristic Algorithm 10 to solve the recourse problem for 15 iterations and in columns of "Relaxed" are reported the results found by using Algorithm 12.

Instances		(PON_{rob})		AARC		Heuristic		Relaxed	
#	$\|V\|$	Time	v	Time	v	Time	v	Time	v
1	5	16	1031	0.07	703	1.2	727	0.8	650
2	5	32	1235	0.06	866	1.3	877	0.8	768
3	5	51	1137	0.07	742	1.3	800	0.8	710
4	5	9	1101	0.07	743	1.3	789	0.7	680
5	5	12	988	0.06	639	1.3	685	0.8	622
6	7	55	1644	0.08	1158	1.8	1121	0.9	1029
7	7	68	1575	0.09	1102	1.8	1105	0.9	987
8	7	55	1577	0.08	1070	1.9	1112	1.0	988
9	9	151	1891	0.1	1250	2.4	1351	1.1	1100
10	9	90	1867	0.11	1255	2.4	1612	1.1	1085

Table 6.1: Comparison of all methods to the optimal value on several instances

The first observation one can make is about the Affinely Adjustable Robust Counterpart, for which we could not decide whether it is a lower or an upper bound to the problem. In practice, its optimal cost is always lower than the optimal one and the case where it gives a greater values was never encountered during our tests. This would mean that AARC gives a lower bound of the problem, even though there is no proof of that. The second observation shows that AARC performs a lot better in terms of solving time compared to all other formulations. However, and that is our third observation, using the heuristic for solving the recourse problem gives a better lower bound than all other methods. Finally, our final observation is to mention that Algorithm 12 performs poorly both considering the solving time and the bound value it provides. Of course, these results have to be tempered since only small graphs are tackled.

Tendencies shown in Table 6.1 were also observed for larger graphs where they were compared using the heuristic Algorithm 10 and randomly generated scenarios (as it is detailed above). Algorithm 12 is always outperformed on both the solving time or the

value it provides by the other two approaches. We managed to solve instances of size 50 with all these approaches in less than an hour. Overall, we observed that even though AARC gives slightly less good solutions than Algorithm 6 combined with Algorithm 10, they were often very close. It happened that the value given by AARC was even better in some rare cases. Regarding the solving time, AARC is always better than the other approaches. Thus, if a choice was to be made on how to generate good first stage solutions, we would recommend to use the AARC approach on the basis of what we tried. Of course, many improvements could be proposed for all the presented methods, and their overall performance in terms of optimal value shows that they are far aways from reaching the true optimal value.

Conclusion

To conclude, we first designed a heuristic for solving the recourse problem, based on the knowledge we could gather at the end of Chapter 5. This heuristic is based on a decomposition of the problem. The graph is splitted in several parts that can be treated separately. For each part, a bi-level problem is posed that we solve heuristically in 3 steps. We first create a sequence of greedy solutions for all bi-level problems, that we improve step by step right after. We first use the maximum stable in a hypergraph to improve low values of the sequence. Then, we use metaedge capacities to improve last values of the sequence. That heuristic was compared to our optimal approach on small instances and its effectiveness was assessed.

In order to solve the problem for larger graphs, we proposed 3 non-exact methods, all based on different relaxations of the problem. Those 3 methods were compared over a large set of instances. So far, results are encouraging, yet not satisfying, especially regarding the optimal value proposed by all methods that are all below the optimal value. Having an upper bound for the problem would be an interesting field to investigate. We now leave the PON deployment problem under demand uncertainty aside to tackle more general and theoretical issues that arise in robust optimization problem.

Algorithm 10 A heuristic for solving the recourse problem $Q(\mathbf{f}, \mathbf{z})$

Require: $\mathbf{f}, \mathbf{z}, G(V, E)$

Decompose the graph G into sub graphs H_p as they are defined by Definition 6.1.1
for $p = 1$ to N_G **do**
  $M_{H_p} \leftarrow \sum_{i \in V_{\mathbf{z}}^{p*}} \mathbf{z}_i$
  $\hat{q}_0^p \leftarrow 0$, $\hat{q}_1^p \leftarrow 1$
  for $u = 2$ to M_{H_p} **do**
    $\hat{q}_u^p \leftarrow 1 + u \times m$
  end for
  Create the hypergraph $\mathfrak{H}_p(V_J^{p*}, E^p)$
  Find $\bar{S}_p$ by solving the problem (PS_p)
  if $|\bar{S}_p| \geq 2$ **then**
    for $u = 2$ to $|\bar{S}_p|$ **do**
      $\hat{q}_u^p \leftarrow u$
    end for
    for $u = |\bar{S}_p| + 1$ to $\left\lfloor \frac{\sum_{j \in \bar{S}_p} d_j^{max}}{m} \right\rfloor$ **do**
      $\hat{q}_u^p \leftarrow |\bar{S}_p| + (u - |\bar{S}_p|) \times m$
    end for
  end if
  for $e_i^p \in E^p$ **do**
    $F_i \leftarrow \sum_{j \in V_d^{p*}} \min\left\{\mathbf{f}_{ij}, d_j^{max}\right\}$
    $F_i^r \leftarrow F_i - \left\lfloor \frac{F_i}{m} \right\rfloor$
  end for
  Sort F_i^r values in a decreasing order
  $\tilde{u}^p \leftarrow \sum_{i \in V_I^{p*}} \left\lfloor \frac{F_i}{m} \right\rfloor$
  if $\tilde{u}^p < M_{H_p}$ **then**
    $\hat{q}_{\tilde{u}^p+1}^p \leftarrow 1 + \tilde{u}^p \times m$
    for $i = 2$ to $M_{H_p} - (\tilde{u}^p + 1)$ **do**
      $\hat{q}_{\tilde{u}^p+i}^p \leftarrow \hat{q}_{\tilde{u}^p+i-1}^p + F_{(i-1)}^r$
    end for
  end if
end for
Solve problem $\hat{Q}_{decomp}(\mathbf{f}, \mathbf{z})$
Combine the results to obtain the final demand vector $\hat{\boldsymbol{d}}$
Solve the problem $\begin{pmatrix} \min \\ (\boldsymbol{\varphi}, \boldsymbol{\zeta}) \in P_G(\hat{\boldsymbol{d}}) \\ \boldsymbol{\zeta} \leq \mathbf{z},\ \boldsymbol{\varphi} \leq \mathbf{f} \end{pmatrix} \sum_{i \in V^*} C\boldsymbol{\zeta}_i$ to obtain $C\hat{\boldsymbol{\zeta}}$ the heuristic worst cost of the recourse problem
Return $\hat{\boldsymbol{d}}$, $C\hat{\boldsymbol{\zeta}}$

Algorithm 11 A heuristic based on the exact framework for solving (PON_{rob})

Require: $LB = -\infty$, L, $\mathfrak{D}_0$, $l = 0$;

1: **while** $l \leq L$ **do**
2: solve PON^l_{rob}; let $\{\mathbf{f}^l, \mathbf{z}^l, \boldsymbol{\gamma}^l\}$ be the solution;
3: $LB \leftarrow v(PON^l_{rob})$;
4: solve $Q_{decomp}(\mathbf{f}^l, \mathbf{z}^l)$ by using Algorithm 10; let $\boldsymbol{d}^{sl}$ be the solution;
5: $\mathfrak{D}_{l+1} \leftarrow \mathfrak{D}_l + \{\boldsymbol{d}^{sl}\}$; $l \leftarrow l + 1$;
6: **end while**

Algorithm 12 Algorithm for solving (PON_{rob}) with continuous relaxation on recourse variables

Require: $LB = -\infty$, $UB = +\infty$, ε, $\mathfrak{D}_0$, $l = 0$

1: **while** $UB - LB > \varepsilon$ **do**
2: solve PON^l_{rob}; let $\{\mathbf{f}^l, \mathbf{z}^l, \boldsymbol{\gamma}^l\}$ be the solution
3: $LB \leftarrow v(PON^l_{rob})$;
4: solve $Q_{relaxed}(\mathbf{f}^l, \mathbf{z}^l)$; let $\boldsymbol{d}^{sl}$ be the solution;
5: $UB \leftarrow \min\{UB, v(Q_{relaxed}(\mathbf{f}, \mathbf{z}))\}$
6: $\mathfrak{D}_{l+1} \leftarrow \mathfrak{D}_l + \{\boldsymbol{d}^{sl}\}$; $l \leftarrow l + 1$
7: **end while**

Algorithm 13 Algorithm for generating arbitrary scenarios in $\mathfrak{D}$

Require: $\bar{d}$ from $\mathfrak{D}$, $S = |\tilde{\mathfrak{D}}|$ the size of the desired sample
Require: $\tilde{V}^*$ containing the nodes of V^* sorted in a pseudo-random order
Require: $P \in [0, 1]$, $\tilde{d}$ a demand vector, d_{pool} an integer
Require: j an integer,(j) being the j-th node of $\tilde{V}^*$

while $S > 0$ **do**
  $d_{pool} \leftarrow \bar{d}$
  $\tilde{d} = 0$
  $j \leftarrow 1$
  while $d_{pool} > 0$ **do**
    Draw a pseudo-random number $r \in [0, 1]$
    if $r \leq P$ and $\tilde{d}_{(j)} < d^{max}_{(j)}$ **then**
      $\tilde{d}_{(j)} \leftarrow \tilde{d}_{(j)} + 1$
      $d_{pool} \leftarrow d_{pool} - 1$
    end if
    if $j = |\tilde{V}^*|$ **then**
      $j \leftarrow 1$
    else
      $j \leftarrow j + 1$
    end if
  end while
  $\tilde{\mathfrak{D}} \leftarrow \tilde{\mathfrak{D}} + \tilde{d}$
  $S \leftarrow S - 1$
end while
return $\tilde{\mathfrak{D}}$

General discussion and conclusion

In order to conclude this work, we shall review our achievements and examine what answer they gave to questions we asked in introduction. For each research axis we explored, we will also propose new investigation field that may arose from our work and that we estimate worth of interest. Except in our last Chapter **??** which tackled a more generic problem, this study focused on the Passive Optical Network deployment optimization problem. This problem has been studied over the past few years, as the need for telecommunication operators to manage that deployment efficiently became a key stake for them. Based on the previous work, we identified several research fields that needed an investigation, the most important being the tackling of demand uncertainty which takes the biggest part of this study. As the literature already furnished the theoretical basis for our research to take place, we took as a main objective to stick as close as possible from real-life expectations in our modeling approaches.

In the chapter 2, we examined the PON deployment problem under the scope of the future cost of the network, which is often more important on the long run. Driven by our objective to be as close as possible from field expectations, we decided to merge our optimization approach with another field: the OA&M (for Organisation, Administration and Management) which, in the telecommunication context, is dedicated to managing all aspects related to the functioning of a telecommunication network. This field, which is, by essence, very engineering oriented provided us with a whole set of possible operational aspects that could be taken into account. As mixing CAPEX and OPEX optimization within the same problem is somewhat unorthodox in operational research, we decided to build a methodology that we tried on two aspects (among others) of the OA&M: the troubleshooting of network failures and the preventive maintenance rounds. For each one of them, we proposed new constraints for our problem that we motivated by being profitable in terms of future costs. In order to assess that profitability, we built cost macro-models that could give a qualitative validation of the constraints we introduced.

The considerations we took into account have been integrated to an operational tool that is used on the field now at Orange. This work also led to a publication in a pair-

reviewed journal [31]. However, even though the operational feedback on this work was positive, there are other OA&M aspects that we did not take into account. As an example, we can quote the management of branching new customers to the network, which is strongly linked to the issue of demand uncertainty. But looking at the problem from a higher point of view, we think that OA&M considerations are not limited to Passive Optical Networks. It is even probable that every network deployment issue implies future costs that have to be taken into account while making the deployment. Therefore, even though our approach remains limited to the field of PON deployment, the methodology we used may be extended to other network deployment problems. In that sense, generalizing this approach to any network design problem would be, in our opinion, scientifically interesting and operationally relevant.

Then, in chapter 3, we examined the Passive Optical Network deployment optimization problem in a tree-graph, on the basis of the preceding work of Kim et al. [34]. We motivated the study of this specific version of the problem by the fact that in the operational tool Gpon Optimizer developed at Orange [22], the problem is decomposed in order to tackle real-size instances and that one step of these decompositions corresponds to the problem studied in this chapter. Therefore, we proposed a new modeling for this problem, from which we derived properties that we used to design a new labeling algorithm for solving this problem for larger instances. We showed that, experimentally, our algorithm outperforms branch-and-bound based approaches from the literature since we were able to solve instances with several thousands of nodes in a few seconds.

Even though our algorithm is efficient, there remains many ways of improving it so it can include other features. For example, the fact that fibers are only allowed to go down the tree can be seen as a limitation. It may be possible to enable fibers to climb the tree for a given number of depth level. Furthermore, limiting the number of splitter sites that may be opened (like in chapter 2) is another feature that could be included in the labeling algorithm we proposed.

In chapter 4, we thus tackled the specific issue of demand uncertainty. Basing our modeling intent on field expectations, we explored several options. We first tried to tackle the problem with a single stage approach for which we established strong limitations that we wanted to overcome. This is why we developed a two-stage robust approach that we modeled so as to stick as much as possible to real-life deployment practices.

Of course, the problem may be posed in another way for another telecommunication operator and it would be very interesting to investigate other variants of the problem. At the end of this chapter, we proposed to make use of available probabilistic information in

order to integrate it within robust approaches. Thus, we proposed a probability bound for estimating the relevance of a given uncertainty set. The last issue on probability bound we tackled in this chapter could be extended to other cases of uncertainty, bounds could be made tighter with the right assumptions, all this in order to increase the confidence that a decision-maker can have in a robust optimization model.

In chapter 5, we made several attempt to solve the two-stage problem introduced in chapter 4 to optimality. To that extent, we designed a column-and-constraint generation algorithm that solves its slave problem by the means of another column-and-constraint generation algorithm. This algorithm proved highly unstable in practice so we made several attempts in order to regulate it by the means of stabilization techniques. Even though the techniques we used are encouraging, they are not fully satisfying and the range of problem we can tackle remains limited in size. However, we were able to withdraw valuable information from solutions of this algorithm.

Solving two-stage robust problem with mixed integer recourse variable really is a challenge and proposing generic methods to tackle these problem in the general case would be a major advance that would benefit the whole field of multi-stage robust optimization. We think that some of the methods we used could be,among other stabilization methods, an inspiration for such work.

In order to improve our solving capacity, we then designed non-exact solving methods in chapter 6. We first designed a heuristic for solving the recourse problem of our robust problem. This heuristic is mostly based on the exploitation of specific problem structures we identified in the previous chapter 5. It especially involves the solving of the maximum stable in a well-chosen hypergraph. We showed that experimentally it was providing good recourse solutions and that it could handle large graphs of more than 500 nodes. Then, based on several relaxations of the problem, we proposed 3 non-exact methods to solve the problem. All these approaches were tested and compared and we could exclude at least on of them which was outperformed by the other 2 on all the criteria we chose.

However, our non-exact approaches are struggling to deal with larger instances. Therefore, based on the information we gathered in this study, it would be interesting to investigate this problem with a method able to tackle it on large graphs. Such method may belong to the field of meta-heuristics, and they would require an extensive research work to be applied to two-stage robust optimization.

To finish, we believe the work presented in the Annex chapter of this work would be worth continuing. In its current state, it is highly theoretical and we lacked the time

to push it to the experimentation. Therefore, the main objective we would recommend to follow is to complete the missing part of the algorithm we designed and to assess its efficiency on several robust problems and, if needed, improving it.

Annex : Probability bound for the recourse cost forecasting in a robust optimization problem with Right-Hand-Side uncertainty

In this section, we present a joint work with Pierre-Louis Poirion[1], currently a Ph.D. student at the CEDRIC laboratory of the CNAM. This work is presented as an annex of the study because it is an ongoing work which needs a little more investigation to be completed. Unfortunately, we lacked the time to finish it properly. However, we estimates that as such, it already provides interesting insights on some generic aspects of robust optimization problems.

To motivate our approach, let us first recall the general modeling of two-stage robust problems with right-hand-side (RHS) uncertainty we aim to deal with, as it is similarly given in definition 1.2.1:

$$(P_{robust}) = \begin{array}{c} \min \\ Cx \geq x \\ x \in X \end{array} \alpha^T x + \max_{d \in D} \begin{array}{c} \min \\ By \geq d - Ax \\ y \geq 0 \end{array} \beta^T y$$

X being a set of vectors belonging to the union of the real positive and natural set (thus making the model valid for any LP, MILP or IP problem with continuous recourse) and D being an uncertainty set defined, without loss of generality, in the positive orthant. The choice of limiting ourselves to RHS uncertainty problems, as it shall be explained later, is due to the fact that first, the time was missing to explore every problem, and second, we identified some interesting properties that theses problems have that could be useful. The inner maximization problem in (P_{robust}) is the recourse problem and is

[1] http://cedric.cnam.fr/index.php/labo/membre/view?id=296
http://uma.ensta-paristech.fr/ poirion?lang=en

denoted by:

$$Q(x, D) = \max_{d \in D} \begin{array}{c} \min \\ By \geq d - Ax \\ y \geq 0 \end{array} \beta^T y$$

The inner minimization problem in $Q(x, D)$ is denoted by:

$$R(x, d) = \begin{array}{c} \min \\ By \geq d - Ax \\ y \geq 0 \end{array} \beta^T y$$

Solving this problem is not the point of this section, as the literature already provides some ways to achieve this. Considering we can solve this problem in a tractable way, there are questions that remains which lead to challenging problems. The latter section addressed the problem of defining the uncertainty set properly. Here, the question we aim to answer is "what are the chances of having a solution that will cost at least *that much*?". In most robust approaches, the worst-case scenario is optimized, but we hardly have any information about what would happen if another scenario is picked. Indeed, we believe that knowing that all other scenarios will cost the same thing as the worst-case scenario or knowing that 90% of scenarios will cost half the worst-case one is an interesting information for decision makers. Of course, this could be done by solving the recourse problem for all possible scenarios, but this is obviously not tractable. Therefore, is it possible, from a given first-stage solution x^* and what is known on d, to estimate the probability of having a cost different from the worst-case one?

The definition and modeling of the problem

Considering a decision-maker to whom a solution x^* to his robust problem (P_{robust}) has been provided, along with a worst possible cost $\Delta^{worst} = \alpha^T x^* + Q(x^*, D)$. However, he may not be satisfied with that worst-case information alone, and he wants to know the chances of having a cost that is either greater or less than a given cost $\Delta \neq \Delta^{worst}$. Therefore, considering d as a random positive (without loss of generality since it is always possible to transform the problem in order to have positive random variables) variable, we aim to find the following probability:

$$\mathcal{P} = Prob\left(\begin{array}{c} \\ \begin{array}{c} \min \\ By \geq d - Ax \\ y \geq 0 \end{array} \beta^T y \leq \Delta - \alpha^T x^* \\ \\ \end{array} \right)$$

without loss of generality since:

$$Prob\left(\begin{array}{c}\min \\ By \geq d - Ax \\ y \geq 0\end{array} \quad \beta^T y \geq \Delta - \alpha^T x^*\right) = 1 - \mathcal{P}$$

Of course, computing this probability as such is not an easy task. It is thus essential to transform this problem into a more convenient one.

Definition 6.2.1. *The set denoted by $\mathfrak{D}_\Delta$ contains all positive vectors d that are such that:*

$$\begin{array}{c}\min \\ By \geq d - Ax^* \\ y \geq 0\end{array} \quad \beta^T y \leq \Delta - \alpha^T x^*$$

Proposition 6.2.2.

$$Q(x^*, \mathfrak{D}_\Delta) \leq \Delta - \alpha^T x^*$$

Proof. This is a direct consequence of the definition 6.2.1 of $\mathfrak{D}_\Delta$, as any $d \in \mathfrak{D}_\Delta$ is such that $R(x, d) \leq \Delta - \alpha^T x^*$, then it is also true for the maximum value this can take. □

Proposition 6.2.3. *For any set D verifying $Q(x^*, D) \leq \Delta - \alpha^T x^*$, then*

$$D \subseteq \mathfrak{D}_\Delta$$

Proof. By definition, as $\mathfrak{D}_\Delta$ contains all vectors verifying $R(x, d) \leq \Delta - \alpha^T x^*$, then it also contains those of D. □

Proposition 6.2.4.

$$\mathcal{P} = Prob\left(d \in \mathfrak{D}_\Delta\right)$$

Proof. It is immediate from propositions 6.2.2 and 6.2.3. □

We thus transformed the problem into another one, which consist in computing the probability of a random vector to belong to a given set. However, there are two remaining issues. For now, we do not have any probabilistic assumption on d and, more importantly, we do not yet know what the set $\mathfrak{D}_\Delta$ looks like.

Theorem 6.2.2. *The set $\mathfrak{D}_\Delta$ verifies:*

$$d \in \mathfrak{D}_\Delta \Leftrightarrow \lambda^{iT} d \leq \Delta - \alpha^T x^* + \lambda^{iT} A x^*, r^j d \leq A x^*, \ \forall i \in I, \forall j \in J$$

where $\left\{\lambda^i | i \in I\right\}$ (respectively $\{r^j, j \in J\}$) is the set of extremal points (respectively the set of extremal rays) of the polyhedron $\left\{\lambda | \lambda^T B \leq \beta\right\}$.

Proof. As given by Proposition 6.2.2, $\mathfrak{D}_\Delta$ is such that $Q(x^*, \mathfrak{D}_\Delta) \leq \Delta - \alpha^T x^*$. Therefore, $d \in \mathfrak{D}_\Delta$ if, and only if

$$R(x^*, d) \leq \Delta - \alpha^T x^*$$

For any d, R can be written as a maximization problem by using its dual counterpart:

$$R(x^*, d) = \min_{\substack{By \geq d - Ax \\ y \geq 0}} \beta^T y = \max_{\substack{\lambda^T B \leq \beta \\ \lambda \geq 0}} (d - Ax^*)^T \lambda$$

Now, let us suppose that $d \in \mathfrak{D}_\Delta$, therefore $R(x^*, d) \leq \Delta_\alpha^T x^*$. Thus, $\max_{\substack{\lambda^T B \leq \beta \\ \lambda \geq 0}} (d - Ax^*)^T \lambda \leq \Delta_\alpha^T x^*$. Considering the polyhedron $\left\{\lambda | \lambda^T B \leq \beta\right\}$, then the inequality must be verified for all its extremal points, which is denoted by I. Therefore,

$$d \in \mathfrak{D}_\Delta \Rightarrow \lambda^{iT} d \leq \Delta - \alpha^T x^* + \lambda^{iT} A x^*, \ \forall i \in I$$

Moreover, since the optimal value of the dual problem is bounded, any extremal ray r^j (of the set of extremal rays J) of the constraint polyhedron must verify $(d - Ax^*).r^j \leq 0$. On the contrary, let us suppose that $d \notin \mathfrak{D}_\Delta$. Therefore $R(x^*, d) > \Delta_\alpha^T x^*$, which means that $\max_{\substack{\lambda^T B \leq \beta \\ \lambda \geq 0}} (d - Ax^*)^T \lambda > \Delta_\alpha^T x^*$. Thus, if the maximization problem is bounded, there must exist an extremal point i' of the polyhedron $\left\{\lambda | \lambda^T B \leq \beta\right\}$ such that $(d - Ax^*)^T \lambda^{i'} > \Delta - \alpha^T x^*$. Hence,

$$d \notin \mathfrak{D}_\Delta \Rightarrow \exists i' : \lambda^{i'T} d > \Delta - \alpha^T x^* + \lambda^{i'T} A x^*$$

If the maximization is unbounded, there exists an extremal ray r^j such that $(d - Ax^*).r^j > 0$. We have proven that $\mathfrak{D}_\Delta$ equals the desired set. □

Corollary 6.2.1. *$\mathfrak{D}_\Delta$ is a bounded polyhedron.*

Proof. This is an immediate consequence of Theorem 6.2.2. □

Now that we showed that $\mathfrak{D}_\Delta$ is a convex bounded polyhedron, let us get deeper in its description.

Definition 6.2.2. *Given two vectors d^1 and d^2, we define a dominance relation between these vectors by stating that d^1 dominate d^2, denoted by $d^1 \succeq d^2$, if, and only if, $d_i^1 \geq d_i^2$ for all $i = 1, ..., |d|$.*

Proposition 6.2.5. *If there exists a vector d that belongs to $\mathfrak{D}_\Delta$, then every vector that is dominated by d also belongs to $\mathfrak{D}_\Delta$.*

Proof. Given $d' \preceq d$, one can assess that $R(x^*, d')$ is a relaxed version of $R(x^*, d)$. Therefore if $R(x^*, d) \leq \Delta_\alpha^T x^*$, this inequality also holds for $R(x^*, d')$. □

Remark 6.2.4. *Note that Proposition 6.2.5 holds even if recourse variables y are integer.*

Inversely, we can give the following proposition:

Proposition 6.2.6. *If there exists a vector d that does **not** belong to $\mathfrak{D}_\Delta$, then every vector that dominate d is **not** in $\mathfrak{D}_\Delta$.*

Proof. Using the same principle as in the proof of the latter proposition, it is immediate. □

Definition 6.2.3. *The hypercube $\mathfrak{H}_\Delta$ is the smallest hypercube such that $\mathfrak{D}_\Delta \subseteq \mathfrak{H}_\Delta$.*

Theorem 6.2.3. *The hypercube $\mathfrak{H}_\Delta$ is defined by the following set of equations:*

$$d \in \mathfrak{H}_\Delta \Leftrightarrow \{d_i \leq d_i^{max}, \forall i = 1, ..., m\}$$

where m is the number of coordinates in vectors $d \in D$ and d_i^{max} is the value of the optimal objective function of the following linear program:

$$(Pd_i) \begin{cases} \max\limits_{d} \; d_i \\ \quad s.t. \;\; \beta^T y \leq \Delta - \alpha^T x^* \\ \qquad\;\; By \geq d - Ax^* \\ \qquad\;\; d_{i'} = 0, \forall i' = 1, .., |D|, \; i' \neq i \\ \qquad\;\; y \geq 0 \end{cases}$$

Proof. First, let us note that values of every d_i component of $d \in \mathfrak{D}_\Delta$ can always be limited by an equation such as $d_i \leq d_i^{max}$, with d_i^{max} being a positive scalar that can be reached by d_i if $d \in \mathfrak{D}_\Delta$. By Proposition 6.2.5, we know that there is at least the intersection of $\mathfrak{D}_\Delta$ with the i-th axis which corresponds to that vector. Therefore, finding d_i^{max} can be done by maximizing d_i with all other components of d set to 0 (or at least free), while ensuring that $R(x^*, d) \leq \Delta - \alpha^T x^*$. That can be done by solving:

$$(Pd_i) \begin{cases} \max\limits_{d} \; d_i \\ \quad \text{s.t.} \;\; R(x^*, d) \leq \Delta - \alpha^T x^* \\ \qquad\;\; d_{i'} = 0, \forall i' = 1, .., |D|, \; i' \neq i \\ \qquad\;\; y \geq 0 \end{cases}$$

As $R(x^*, d)$ is a minimization problem in y, and (Pd_i) is a maximization problem in d, the minimization can be removed from the constraint in order to obtain a linear program. □

Remark 6.2.5. *Note that the ratio between the volume of* $\mathfrak{D}_\Delta$ *and* $\mathfrak{H}_\Delta$ *is bounded:*

$$\frac{1}{2} \leq \frac{V(\mathfrak{D}_\Delta)}{V(\mathfrak{H}_\Delta)} \leq 1$$

Proof. This is a direct consequence of how $\mathfrak{H}_\Delta$ is built in Theorem 6.2.3 and the property given by proposition 6.2.5. Indeed, using the same considerations as shown in the proof of the latter theorem, one can show that hyperpyramid pointed towards the origin that is the half of $\mathfrak{H}_\Delta$ will always be contained by $\mathfrak{D}_\Delta$. □

Figure 6.6 gives an illustrative example of how a given set $\mathfrak{D}_\Delta$ is contained in $\mathfrak{H}_\Delta$ and how it will always contain its half.

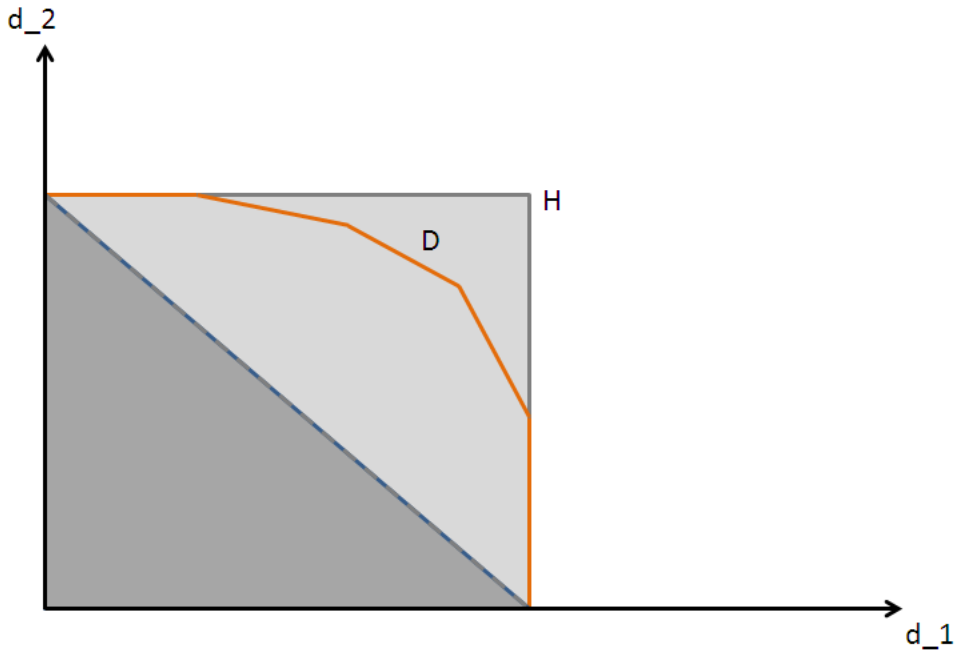

Figure 6.6: Illustrative example of a set $\mathfrak{D}_\Delta$ (in Orange) that is contained in the corresponding set $\mathfrak{H}_\Delta$

Illustrative example of finding $\mathfrak{D}_\Delta$

In order to illustrate the claims of the latter section, let us show how this work on a small example. Let us consider the following problem:

$$\begin{aligned} R_{ex}(x^*, d) = \min_{y_1, y_2} \quad & y_1 + 2y_2 \\ \text{s.t.} \quad & y_1 + y_2 \geq d_1 - x^* \\ & y_1 + 3y_2 \geq d_2 - x^* \\ & y_1, y_2 \geq 0 \end{aligned}$$

We thus aim to find the set $\mathfrak{D}_\Delta$, associated to a cost Δ, which is such that every $(d_1, d_2) \in \mathfrak{D}_\Delta$ makes $R(x^*, d) \leq \Delta - x^*$ (here, $\alpha = 1$), for positive values of d_1 and d_2. By dualizing R, we get:

$$\begin{aligned} R_{ex}(x^*, d) = \max_{\lambda_1, \lambda_2} \quad & (d_1 - x^*)\lambda_1 + (d_2 - 2x^*)\lambda_2 \\ \text{s.t.} \quad & \lambda_1 + \lambda_2 \leq 1 \\ & \lambda_1 + 3\lambda_2 \leq 2 \\ & \lambda_1, \lambda_2 \geq 0 \end{aligned}$$

The extremal points of the latter linear program's polyhedron are: $(0,0)$, $(1,0)$, $\left(0, \frac{2}{3}\right)$, $\left(\frac{1}{2}, \frac{1}{2}\right)$.

Therefore, by Theorem 6.2.2,

$$(d_1, d_2) \in \mathfrak{D}_\Delta \Leftrightarrow \begin{cases} d_1 \leq \Delta \\ d_2 \leq \frac{3}{2}\Delta + \frac{1}{2}x^* \\ d_1 + d_2 \leq 2\Delta \\ d_1, d_2 \geq 0 \end{cases}$$

which is true for any value of Δ and x^*, as long as $x^* \leq \Delta$, which is given by the extremal point $(0,0)$. Figure 6.7 gives a graphic representation of $\mathfrak{D}_\Delta$ for different values of Δ and x^*.

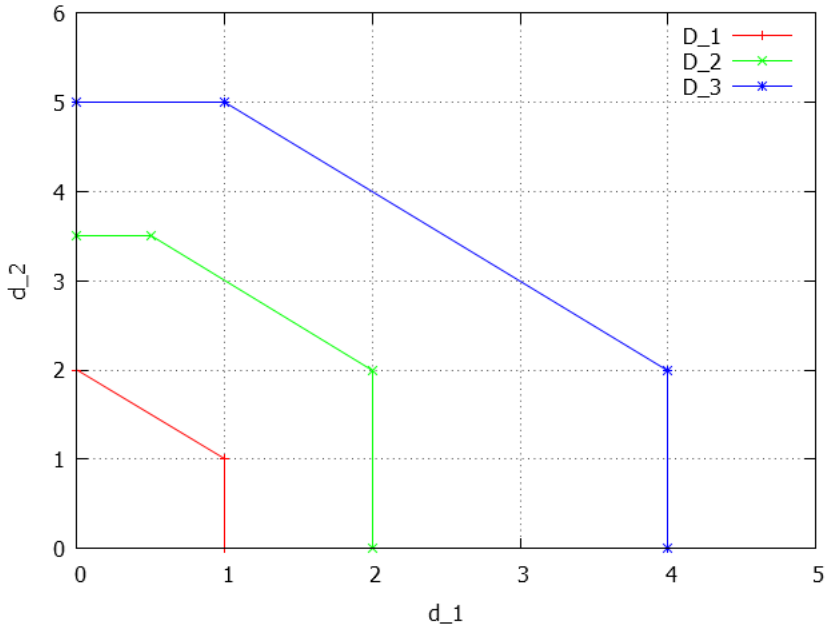

Figure 6.7: Graphic representation of $\mathfrak{D}_1$, $\mathfrak{D}_2$ and $\mathfrak{D}_3$ for $x^* = 1$

Now that we are able to find, theoretically, the set $\mathfrak{D}_\Delta$, the last part remains, that is how to compute the probability.

Properties and computation feasibility of the probability $\mathcal{P}$

In practice, as the actual problem R may be wide in size, the size of the polyhedron $\left\{\lambda | \lambda^T B \leq \beta\right\}$ may be wide as well. Therefore, enumerating all it extremal points in order to have a full description of $\mathfrak{D}_\Delta$ could prove hardly intractable in some cases. Moreover, even having that description would not help directly to find the probability. First, let us postulate the following hypothesis:

Hypothesis 5. *Uncertain variables d_i that compose the uncertain vector d are all independent and follow a known probability law such that $Prob(d_i \leq a)$ can be computed for any a value.*

For this hypothesis, let us provide theorems that allows to compute both lower and upper bounds for the probability in a specific case.

Definition 6.2.4. *Given a vector $\bar{d}$, we denote by $\mathfrak{H}_{\bar{d}}$ the hypercube defined by the following set of equations:*

$$d \in \mathfrak{H}_{\bar{d}} \Leftrightarrow 0 \leq d_i \leq \bar{d}_i, \, \forall i$$

Proposition 6.2.7. *Given a vector $\bar{d}$, under hypothesis 5, we have:*

$$Prob(d \in \mathfrak{H}_{\bar{d}}) = \prod_i \left(Prob(d_i \leq \bar{d}_i) - Prob(d_i \leq 0)\right)$$

Proof. For any vector d,

$$Prob(d \in \mathfrak{H}_{\bar{d}}) = Prob(0 \leq d_i \leq \bar{d}_i, \, \forall i)$$

Under hypothesis 5, d_i random variables are supposed to be independent. Therefore,

$$\begin{aligned} Prob(d \in \mathfrak{H}_{\bar{d}}) &= Prob(\bigcap_i 0 \leq d_i \leq \bar{d}_i) \\ &= \prod_i Prob(0 \leq d_i \leq \bar{d}_i) \\ &= \prod_i \left(Prob(d_i \leq \bar{d}_i) - Prob(d_i \leq 0)\right) \end{aligned}$$

□

Note that $\left(Prob(d_i \leq \bar{d}_i) - Prob(d_i \leq 0)\right)$ can always be computed under hypothesis 5, thus making the computation of $Prob(d \in \mathfrak{H}_{\bar{d}})$ possible as well.

Theorem 6.2.4. *Under hypothesis 5, if $\exists d' \in \mathfrak{D}_\Delta$ and $\exists d'' \in \mathfrak{H}_\Delta \setminus \mathfrak{D}_\Delta$, then*

$$Prob\left(d \in \mathfrak{H}_{d'}\right) \leq \mathcal{P} \leq Prob\left(d \in \mathfrak{H}_\Delta\right) - Prob\left(d'' \leq d \leq d^{max}\right)$$

with d_i^{max} values being those defined with $\mathfrak{H}_\Delta$.

Proof. As it is said by proposition 6.2.5, if $d' \in \mathfrak{D}_\Delta$, all vectors d dominated by d' are also in $\mathfrak{D}_\Delta$. $\mathcal{P}$ is thus superior to the probability of d belonging to $\mathfrak{H}_{d'}$ and, by definition, inferior to the probability of d belonging to $\mathfrak{H}_\Delta$ minus the probability to fall into the hypercube defined by d'' and d^{max}. □

The principle here is that finding a single d' that belongs to $\mathfrak{D}_\Delta$ enables one to find a subset of $\mathfrak{D}_\Delta$ for which the belonging probability is easy to compute and gives a lower bound. Of course, if this is true for a single point d' (see figure 6.8), it is also true if several vectors in $\mathfrak{D}_\Delta$ are found, thus making the bound better (as in figure 6.9). In both figures, the grey area is the set used to approximate the complete $\mathfrak{D}_\Delta$ which is depicted by the orange lines.

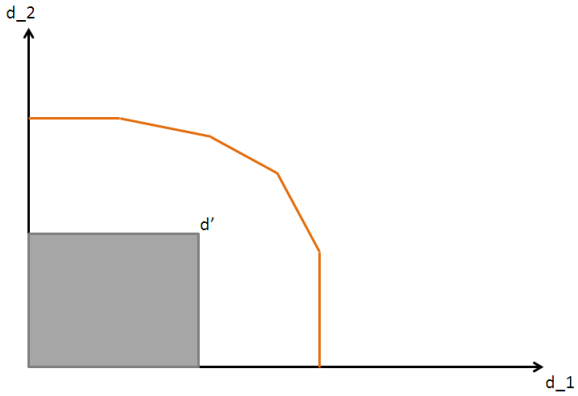

Figure 6.8: Illustration for a single d' found in $\mathfrak{D}_\Delta$

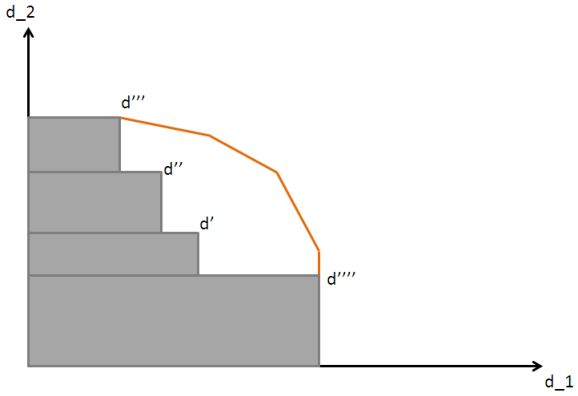

Figure 6.9: Illustration for several vectors found in $\mathfrak{D}_\Delta$

Theorem 6.2.5. *Given several vectors d^k that do not dominate each other, for $k = 1, ..., K$ with $K \leq |D|$, such that $d^k \in \mathfrak{D}_\Delta$, under hypothesis 5, we have:*

$$Prob\left(\bigcup_k d \in \mathfrak{H}_{d^k}\right) \leq \mathcal{P}$$

Proof. This is an immediate generalization of the lower bound of theorem 6.2.4 for several valid vectors. □

Proposition 6.2.8. *The decision problem that consists in determining if a given vector d does belong or not to $\mathfrak{D}_\Delta$ is polynomially solvable.*

Proof. It is immediate since one just needs to solve $R(x^*, d)$, which is a continuous linear program, and compare it to $\Delta - \alpha^T x^*$. □

Using the same principle as for Theorems 6.2.4 and 6.2.5, we can propose the following theorem to reduce the upper bound proposed in those theorems.

Theorem 6.2.6. *Under hypothesis 5, supposing that* $\exists d_i^k, \forall k = 1, ..., K \leq |D|$ *that do not dominate each other, such that* $d^k \in \mathfrak{H}_\Delta \setminus \mathfrak{D}_\Delta, \forall k = 1, ..., K$*, we have*

$$\mathcal{P} \leq Prob(d \in \mathfrak{H}_\Delta) - Prob(\bigcup_k d^k \leq d \leq d^{max})$$

Proof. This is an immediate generalization of the lower upper bound of theorem 6.2.4, considering that we must remove the probability for d to fall into the union of several hypercubes from the probability for d to fall in $\mathfrak{H}_\Delta$. □

At the time this study was written, the work presented in this section was still ongoing. Therefore, the computation of the probability of the vector d to belong to a union of hypercubes $\mathfrak{H}_{\bar{d}}$ is the missing part of the algorithm presented in the next section. We think this computation is possible for a little number of hypercubes but, due to time limitations, we were not able to fulfill this task. We thus leave that aspect of the problem as an opened research way that would allow one to apply these results.

Development of a generic algorithm for probability approximation computation

As it is described in the previous section 6.2.4, we can give an upper bound and a lower bound to the probability we aim to find here, under two kinds of hypothesis, by approximating a polyhedron $\mathfrak{D}_\Delta$. This is done by finding K vectors d in $\mathfrak{H}_\Delta$ and, after finding if they also belong to $\mathfrak{D}_\Delta$ (which is done in polynomial time, see Proposition 6.2.8). As one could expect, the quality of the approximation will greatly depend on the set of the chosen d^k vectors. Therefore, that choice alone is an issue that may be tackled in various ways. Lots of work are necessary for the development and testing of all these approaches and for this Ph.D. thesis, time was lacking to complete it. However, we believe this would be an interesting research field and we would like to expose the few methods and strategies we could think about in order to find a good approximation for $\mathcal{P}$. To that extent, most methods presented in this subsection will not be highly detailed, as we only focus on the principle that lies behind it. First, let us describe the two generic framework one could work with while trying to approximate $\mathcal{P}$.

Algorithms 14 and 15 are very similar in appearance. What will differ is that in one case, d^k vectors are found before computing $\mathcal{P}$'s lower and upper bounds, whereas in the other case, d^k vectors are added until the lower and upper bounds are close enough. Therefore, in both cases, the core of the algorithm will be the actual finding of those vectors (it correspond to the step 3 of Algorithm 14, and step 5 of Algorithm 15).

Algorithm 14 Computation of an approximation of $\mathcal{P}$ for a fixed K value

Require: K, Δ, x^*, $R(x^*, d)$
Require: S_D the set of vectors that do belong to $\mathfrak{D}_\Delta$
Require: S_H the set of vectors that do belong to $\mathfrak{H}_\Delta \setminus \mathfrak{D}_\Delta$
Require: $LB_\mathcal{P}$, $UB_\mathcal{P}$
$S_D \leftarrow \emptyset$, $S_H \leftarrow \emptyset$
Define $\mathfrak{H}_\Delta$ using Theorem 6.2.3
Find K vectors d^k that do belong to $\mathfrak{H}_\Delta$
for $k = 1$**to** $k = K$ **do**
  if $d^k \in \mathfrak{D}_\Delta$ **then**
    $S_D \leftarrow S_D + d^k$
  else
    $S_H \leftarrow S_H + d^k$
  end if
end for
Compute $LB_\mathcal{P}$ using S_D and Theorem 6.2.5
Compute $UB_\mathcal{P}$ using S_H and Theorem 6.2.6
return $LB_\mathcal{P}$, $UB_\mathcal{P}$

Depending on the effort one is ready to put into finding "good" vectors, these procedures may require more or less computing. Indeed, for example, vectors that are pointing to the facets of $\mathfrak{D}_\Delta$ will be the most interesting for computing the lower bound. Therefore, we listed a few d-finding strategies that could be interesting for this problem:

- Generate d vectors randomly within $\mathfrak{H}_\Delta$, which could prove useful for cases where computing $R(x^*, d)$ is really fast and the time lost on evaluating "bad" d vectors is negligible.
- Define a given number of rays from the origin to a given direction and approach its intersection with $\mathfrak{D}_\Delta$ by dichotomy with a given precision (which can be seen as a more precise random approach).
- Use Theorem 6.2.2 to generate a subset of facets of $\mathfrak{D}_\Delta$ by finding a few extremal points of the polyhedron $\left\{\lambda | \lambda^T B \leq \beta\right\}$ and choose d vectors exclusively on these facets.

Of course, other approaches may be found, and their use will highly depend on the problem the decision-maker is dealing with. Therefore, as time was lacking to explore all these options, we chose not to investigate them in depth. Instead, we preferred to assess the interest of our approach by applying it to a simple two-stage robust problem.

Algorithm 15 Computation of an approximation of $\mathcal{P}$ with a fixed precision ϵ

Require: ϵ, Δ, x^*, $R(x^*, d)$
Require: S_D the set of vectors that do belong to $\mathfrak{D}_\Delta$
Require: S_H the set of vectors that do belong to $\mathfrak{H}_\Delta \setminus \mathfrak{D}_\Delta$
Require: $LB_\mathcal{P}$, $UB_\mathcal{P}$
$S_D \leftarrow \emptyset$, $S_H \leftarrow \emptyset$
$LB_\mathcal{P} \leftarrow 0$, $UB_\mathcal{P} \leftarrow 1$
Define $\mathfrak{H}_\Delta$ using Theorem 6.2.3
while $UB_\mathcal{P} - LB_\mathcal{P} > \epsilon$ **do**
  Find a vector $d \in \mathfrak{H}_\Delta$
  if $d \in \mathfrak{D}_\Delta$ **then**
    $S_D \leftarrow S_D + d$
    Compute $LB_\mathcal{P}$ using S_D and Theorem 6.2.5
  else
    $S_H \leftarrow S_H + d$
    Compute $UB_\mathcal{P}$ using S_H and Theorem 6.2.6
  end if
end while
return $LB_\mathcal{P}$, $UB_\mathcal{P}$

Conclusion

To conclude, this chapter investigated two different problems linked to data uncertainty. We first proposed a way to compute a bound on the probability for a random vector to belong in a given uncertainty set, under few probabilistic assumptions. This led to the minimization of a single variable function for which we were able to give an analytic solution in an ideal case. For more general cases, we showed that a simple numeric approach could help to find that minimum with very high precision in a few seconds. We applied these results to assess how useful the bound can be. The main use for this bound is for practical cases where the few number of probabilistic assumptions we made are true, thus enabling one to build an uncertainty set with an *a priori* bounded validation from the bound.

In a second time, we examined the robust cost of a two-stage robust problem with right-hand-side uncertainty. We developed theoretical tools to describe the problem and its implications. Under few probabilistic assumptions on data uncertainty, we especially showed that the probability for the recourse cost to be under a given value is equivalent to the probability for the uncertain data vector to belong to a bounded polyhedron for which we gave an extensive description. From there, we proposed a generic algorithm

to compute a lower and an upper bound on that probability. However, due to time limitations, we were not able to develop an essential part of this algorithm, thus leaving an opened research field to investigate. Even though it is still early to claim that our approach really can answer the problem we tackled, we motivated it by our will to answer one of robust optimization's drawbacks: the fact that the only relevant scenario is the worst one. We estimate decision makers may want to obtain other information from a robust solution, and our approach may be a part of it.

Bibliography

[1] A. Atamtürk., M. Zhang. *Two-stage robust network flow and design under demand uncertainty*, Operation Research, Vol. 55, pp 662-673, 2007.

[2] F. Babonneau., J.P. Vial., R. Apparigliato. *Handbook on "Uncertainty and Environmental Decision Making"*, chapter Robust Optimization for environmental and energy planning, International Series in Operations Research and Management Science, Springer Verlag, 2009.

[3] F. Babonneau., O. Klopfenstein, A. Ouorou., J.P. Vial. *Robust capacity expansion solutions for telecommunications networks with uncertain demands*, Technical paper, Orange Labs R& D, 2009.

[4] W. Ben-Ameur., H. Kerivin. *Routing of Uncertain Traffic Demands*, Optimization and Engineering Vol. 6, 2005, pp 283-313.

[5] W. Ben-Ameur., M. Zotkiewicz. *Robust routing and optimal partitioning of a traffic demand polytope*, International Transactions in Operational Research, Vol. 18(3), pp 307–333, 2011.

[6] A. Ben-Tal., A. Goryashko., E. Guslitzer., A. Nemirovski. *Adjustable robust solutions of uncertain linear programs*, Mathematical Programming, Vol. 99, pp. 351-376, 2004.

[7] A. Ben-Tal., L. El-Ghaoui., A. Nemirovski. *Robust Optimization*, Princeton Series in Applied Mathematics, 2006.

[8] C. Berge. *Graphs and Hypergraphs*, North-Holland Publishing Company, 1973.

[9] D. Bertsimas., D.B. Brown., C. Caramanis. *Theory and Applications of Robust Optimization*, SIAM Review, Vol. 53(3), pp. 464–501, 2011.

[10] D. Bertsimas., C. Caramanis. *Finite Adaptability in Multistage Linear Optimization*, IEEE Transactions on Automatic Control, Vol. 55 (2), pp. 2751–2766, 2010.

[11] D. Bertsimas., V. Goyal. *On the Power of Robust Solutions in Two-Stage Stochastic and Adaptive Optimization Problems*, Mathematics of Operations Research, Vol. 35(2), pp. 234–305, 2010.

[12] D. Bertsimas., V. Goyal. *On the Power and Limitations of Affine Policies in Two-Stage Adaptive Optimization*, Mathematical Programming, Vol. 134(2), pp. 491–531, 2012.

[13] D. Bertsimas., D.A. Iancu., P.A. Parrilo. *Optimality of Affine Policies in Multistage Robust Optimization*, Mathematics of Operations Research, Vol. 35(2), pp. 363–394, 2010.

[14] D. Bertsimas., M. Sim. *Robust Discrete Optimization and Network Flows*, Mathematical Programming Series B, Vol. 98, 2003.

[15] D. Bertsimas., M. Sim. *The Price of Robustness*, Operations Research Vol. 52, No. 1, January-February 2004, pp. 35-53.

[16] A. Billionnet. *Optimisation Discrète : De la modélisation à la résolution par des logiciels de programmation mathématique*, Dunod, série InfoPro, 2006.

[17] A. Billionnet., M.C. Costa., P.L. Poirion. *2-Stage Robust MILP with continuous recourse variables*, submitted, 2013.

[18] J.R. Birge., F. Louveaux. *Introduction to Stochastic Programming*, Springer Series in Operations Research and Financial Engineering, 1997.

[19] R. Boutaba., J. Xiao. *Network Management: State of the Art*, Proceedings of the World Computer Congress, pp. 127–146, 2002.

[20] B. Büeler., A. Enge., K. Fukuda. *Exact Volume Computation for Polytopes: A Practical Study*, Polytopes - Combinatorics and Computation, DMV Seminar Vol. 29, pp 131–154, 2000.

[21] M. Chardy., M.C. Costa., A. Faye., M. Trampont. *Optimizing splitter and fiber location in a multilevel optical FTTH network*, European Journal of Operational Research, Vol. 222, pp 430–440, 2012.

[22] M. Chardy., M.C. Costa., A. Faye., S. Francfort., C. Hervet., M. Trampont. *La RO au coeur du déploiement du Fiber To The Home à France-Télécom Orange : La RO récompensée par le prix Orange de l'Innovation 2012 (catégorie Réseau)*, Bulletin de la ROADEF, Article Invité, Vol. 29, pp. 8-11, 2012.

[23] G.B. Dantzig. *Linear Programming under Uncertainty*, Management Science, Vol. 1 (3-4), 1955.

[24] S. Dempe. *Bilevel Programming - A Survey*, Discrete Bilevel Optimization Problems, Chapter 1, 1996.

[25] V. Gabrel., C. Murat. *Robustness and duality in linear programming*, Journal of the Operational Research Society, Vol. 61, pp 1288-1296, 2009.

[26] V. Gabrel., C. Murat., N. Remli. *Linear Programming with interval right hand sides*, International Transactions in Operational Research, Vol. 17, pp 397-408, 2010.

[27] L. Galand. *Méthodes exactes pour l'optimisation multicritère dans les graphes : recherche de solutions de compromis*. Ph.D. thesis (in French), University of Paris VI, 2008.

[28] M.R. Garey., D.S. Johnson. *Computers and Intractability, A guide to the theory of NP-Completeness*, W.H.Freeman, San Francisco, 1979.

[29] S. Gollowitzer., L. Gouveia., I. Ljubic. *The Two Level Network Design Problem with Secondary Hop Constraints*, Proceedings of the 5th International Network Optimization Conference (INOC 2011), 71–76.

[30] S. Gollowitzer., I. Ljubic. *MIP models for connected facility location: a theoretical and computational study*, Computers & Operations Research **38**(2) (2011), 435–449.

[31] C. Hervet., M. Chardy. *Passive optical network design under operations administration and maintenance considerations*, Journal of Applied Operational Research, Vol. 4(3), pp. 152-172, 2012.

[32] D.A. Iancu., M. Sharma., M. Sviridenko. *Supermodularity and Affine Policies in Dynamic Robust Optimization*, Operations Research, A VOIR.

[33] A. Jarray, B. Jaumard, A.C. Houle. *Minimum CAPEX/OPEX Design of Optical Backbone Networks*, In: International Conference on Ultra Modern Telecommunications & Workshops, (2009)

[34] Kim, Y., Lee, Y., L. and Han, J.: A splitter location-allocation problem in designing fiber optic access networks. In: European Journal of Operational Research, Vol 210, Issue 2, pp. 425–435, (2011)

[35] P. Kouvelis., G. Yu. *Robust Discrete Optimization and Its Applications*, Kluwer Academic Publishers, (1997).

[36] Lee, Y., Kim, Y. and Han, J.: FTTH-PON Splitter Location-Allocation Problem. In: Proceedings of the eighth INFORMS Telecommunications Conference, (2006)

[37] T.L. Magnanti., R.T. Wong. *Accelerating Benders decomposition: algorithmic enhancements and model selection criteria*, In: Operations Research, Vol. 29, pp. 464–484, 1981.

[38] M. Minoux. *Robust linear programming with right-hand-side uncertainty, duality and applications*, L.A. Floudas., P.M. Pardalos. (eds) Encyclopedia of Optimization, 2nd edn., pp. 3317-3327, 2008.

[39] M. Minoux. *Robust network optimization under polyhedral demand uncertainty is NP-hard*, Discrete Applied Mathematics, 158, pp. 597-603, 2010.

[40] M. Minoux. *On 2-stage robust LP with RHS uncertainty: complexity results and applications*, J Glob Optim, Vol. 49, pp 521-537, 2011.

[41] I. Newton. *De analysi per aequationes numero terminorum infinitas*, 1711.

[42] W. Ogryczak., T. Sliwinski. *On solving linear programs with the ordered weighted averaging objective*, In: European Journal of Operational Research, Vol. 148, pp. 80–91, 2003.

[43] A. Ouorou. *Tractable approximations to a robust capacity assignment model in telecommunications under demand uncertainty*, under revision.

[44] G. Polya., G. Szegõ. *Aufgaben und Lehrsätze der Analysis [Problems and Theorems in Analysis]*, Berlin: Springer-Verlag, 1925.

[45] T.M. Range. *An Integer Cutting-Plane Procedure for the Dantzig-Wolfe Decomposition: Theory*, Discussion Papers on Business and Economics 10, 2006.

[46] N. Remli. *Robustesse en Programmation Linéaire*, LAMSADE (Université Paris-Dauphine), thèse soutenue le 17 Mars 2011.

[47] A.L. Soyster. *Convex programming with set-inclusive constraints and applications to inexact linear programming*, Operation Research, Vol. 21, pp 1154-1157, 1973.

[48] S. Takriti., S. Ahmed. *On robust optimization of two-stage systems*, Mathematical Programming Vol. 99, Issue 1, 2004, pp. 109-126.

[49] A. Thiele., T. Terry., M. Epelman. *Robust Linear Optimization With Recourse*, 2010.

[50] M. Trampont. *Modélisation et optimisation du déploiement des réseaux de télécommunications: application aux réseaux d'accès cuivre et optique.*, Ph.D Laboratoire CEDRIC, octobre 2009.

[51] F. Vanderbeck. *On Dantzig-Wolfe Decomposition in Integer Programming and Ways to Perform Branching in a Branch-and-Price Algorithm*, Operations Research Vol. 48, No. 1, January-February 2000, pp. 111-128.

[52] F. Vanderbeck.,M.W.P. Savelsbergh. *A generic view of Dantzig-Wolfe decomposition in mixed integer programming*, Operations Research Letters 34, pp. 296-306, 2006.

[53] Q. Wang Q., W. Li. *Solution on OAM of FTTH based on PON*, In: Information & Communications, 2009.

[54] B. Zeng., L. Zhao. *Solving two-stage robust optimisation problems using a column-and-constraint generation method*, Operations Research Letters 41(5), pp. 457-461, 2013.

[55] B. Zeng., L. Zhao. *An Exact Algorithm for Two-stage Robust Optimization with Mixed Integer Recourse Problems*, Optimization Online, 2012.