Contributions à la résolution de problèmes d'optimisation combinatoire difficiles

Zacharie Alès

INSTITUT POLYTECHNIQUE DE PARIS

ENSTA

IP PARIS

Mémoire d'Habilitation à Diriger des Recherches

Contributions to solving hard combinatorial optimization problems

Mémoire d'Habilitation à Diriger des Recherches
de l'Institut Polytechnique de Paris

préparée à l'UMA, ENSTA Paris, Institut Polytechnique de Paris

ZACHARIE ALES

Composition du Jury :

Prof.	Stéphane Canu	LITIS, INSA Rouen, France	Examinateur
D.R.	Claudia Dambrosio	LIX, Ecole Polytechnique, France	Examinateur
Prof.	Sourour Elloumi	UMA, ENSTA Paris, France	Examinateur
Prof.	Bernard Fortz	HEC Liège, Belgique	Rapporteur
Prof.	Pierre Fouilhoux	LIPN, Université Sorbonne Paris Nord, France	Rapporteur
Prof.	Marco Lübbecke	Université RWTH D'Aachen, Allemagne	Rapporteur
M.C.F.	Jose Neto	SAMOVAR, Telecom SudParis, France	Examinateur

Contents

Chapter 1

Introduction

1.1 Context

A Mixed Integer Linear Program (MILP) is a mathematical model that involves optimizing a linear function over a feasible region described by linear constraints where a subset of variables are restricted to integer values.

The study of MILPs is a very active field of research that encompasses the solution of a diverse range of concrete problems such as energy production, railway scheduling and warehouses location. This thesis is organized into three chapters, each dedicated to a line of research I have undertaken in order to define and solve MILPs. My objective is to provide solutions that not only have good objective values but are also satisfactory in terms of their practical applicability for real-world problems.

In the context of this thesis, a *formulation* corresponds to a MILP mathematical model of a combinatorial optimization problem. For a given problem, multiple formulations with distinct variables or constraints may be considered. In this thesis, various examples illustrate the significant impact that the choice of a formulation can have on the numerical performances. Solving a combinatorial optimization problem optimally can be achieved by providing a formulation to generic solvers such as Scip [1], CPLEX [38] or Gurobi [63]. Despite mixed integer linear programming being $\mathcal{NP}$-hard [72], these solvers have made impressive progress in recent decades. However, when problems size increases, many of them quickly become computationally intractable. My first line of research is dedicated to overcoming this limitation in two complementary directions:

1. improving the problem modeling: extended formulations, breaking symmetries, identifying stronger alternative models, etc.;
2. enhancing the solution method: exact decomposition algorithms (e.g., column generation, Benders decomposition), parameters selection, dynamic addition of constraints during the resolution, algorithms with performance guarantees, tailored heuristics, etc.

Various methods can be considered to obtain MILP solutions with good objective values. Nonetheless, this is not necessarily sufficient to guarantee that the solution can be implemented in practice. Indeed, when solving a problem it is generally unrealistic to assume precise knowledge of all its objective and constraints coefficients. For example, travel time depend on road traffic and energy output derived from solar panels are influenced by the level of sunlight. These values may be hard to determine precisely in advance. Consequently, an optimal solution for a specific value of the coefficients can rapidly deteriorate or even become infeasible when the coefficients are subject to minor perturbations, thus making it invalid from a practical point of view. Henceforth, my second line of research is to take into account uncertainty in MILPs. Two main approaches have been considered for this purpose:

1. *stochastic optimization* in which it is assumed that the uncertain coefficients vary according to probability distributions; and

2. *robust optimization* in which we seek a solution with the best objective value for the worst possible value of the uncertain coefficients.

The choice of the approach depends on the problem under consideration. If the aim is to obtain solutions that perform well on average, stochastic optimization should then be preferred. Conversely, if the aim is to strongly avoid the most unfavorable scenarios (e.g., when human lives are at stake), robust optimization appears to be a more suitable alternative.

My final area of research to efficiently solve MILPs is focused on the use of data. It is estimated that more than 10^{18} bytes of data are generated every day. Hence, when confronted with a specific real-life application, the collection of a substantial volume of potentially valuable data becomes possible. This has contributed to a huge surge in the development of Machine Learning (ML) methods in recent years. As a consequence many approaches have been developed to use ML methods to help solve operational research problems, and vice versa. To improve the definition and the solution of MILPs, various objectives have been considered such as learning how to branch [62], selecting relevant valid inequalities to add during the resolution [92], or defining appropriate uncertainty sets in robust optimization [104]. Other works focus on solving MILPs to improve the resolution of Machine Learning (ML) problems such as learning interpretable classifiers [3, 24] or identifying adversarial examples [53].

In the following, I summarize the main contributions presented in this thesis.

1.2 Main contributions

In the course of my research career, I had the opportunity to study diverse combinatorial optimization problems in collaboration with colleagues from the research laboratories to which I have belonged. For example, I worked on the identification of recurrent dialogical patterns during my PhD thesis at INSA Rouen [P9, P10], on the correlation clustering and scheduling problems at Laboratoire d'Informatique d'Avignon [P3, P8, P17] and on the placement of virtual machines at the Unité de Mathématiques Appliquées d'ENSTA Paris [P16].

In this thesis I present my main contributions in the three lines of research described in the previous section.

1.2.1 Improving the solution of hard MILPs

A consequent part of my research focuses on improving the modeling and the resolution of combinatorial optimization problems [P1, P3, P5–P7, P11, P16, P17]. Chapter 2 addresses two of these contributions.

The problem of identifying dialogical patterns considered during my PhD thesis led us to consider the K-partitioning problem which consists in partitioning an edge-weighted graph into $K \in \mathbb{N}^*$ clusters such that the sum of the weights inside the clusters is minimized. We proposed for this problem a new formulation as well as a cutting plane algorithm [P7]. Section 2.1 presents our subsequent work on that topic. We first extend our formulation by the addition of edge variables and show that it enables to strengthen the linear relaxation [P6]. Then we identify a new family of valid inequalities involving these new variables and determine conditions under which this family and three others are facet-defining of the extended formulation. We introduce an efficient branch-and-cut algorithm based on the generation of facet-defining inequalities that provides better numerical results than the direct resolution of our two formulations and two generic clustering formulations from the literature that we adapt to the K-partitioning problem. Indeed, among all the instances considered from the TSPLib [95], 34% of them are solved to optimality only by our branch-and-cut algorithm. Finally, we provide bounds on the linear relaxation of the two formulations from the literature which could explain their poor numerical performances.

We then consider in Section 2.2 a classical location problem called the p-median problem [P11] that we studied in the context of the PhD thesis of Cristian Duran. This problem considers a set of clients, a set of sites, and distances between the clients and the sites. The objective is to select p sites to open such that the sum of the distances between each client and its closest opened site is minimized. We first consider a Benders decomposition for this problem and introduce

an efficient algorithm to separate the Benders cuts. Secondly, we propose a two-step algorithm based on this decomposition to solve the problem. In its first step, the integrality of the variables is relaxed which enables to quickly provide strong bounds on the optimal solution. Finally, we perform an extensive computational study of more than 250 instances from 6 distinct datasets from the literature. This experiment shows that our algorithm outperforms the other state of the art methods. Among all the instances which optimal solution was previously unknown, we are able to improve the best known solution for 91% of them and to guarantee their optimality in 81% of the cases.

1.2.2 Dealing with uncertainty

My second line of research, presented in Chapter 3, is robust optimization [P2, P8, P14].

In a previous work we considered the solution of the deterministic p-center problem for which we proposed two new formulations [P1]. This discrete location problem is similar to the p-median problem except that the objective is to minimize the maximal distance between any client and its closest opened site. In Section 3.1 we present a new a robust variant of this problem, studied during the PhD thesis of Cristian Duran, in which the uncertainty lies in the value of the node weights and of the distances [P12]. By assumption they can take any value in a box-uncertainty set. In this two-stage problem the sites to open must be decided before the uncertainty is revealed while the allocation of the clients to the opened sites is determined afterwards. Our most important result is that considering a finite number of uncertain scenarios is sufficient to obtain an optimal solution. We use this result to adapt five different formulations of the deterministic weighted p-center problem to our robust variant. Finally, we define and numerically evaluate two exact algorithms based on these formulations: a column-and-constraint generation algorithm and a branch-and-cut algorithm. The latter can only be used with two robust formulations which deterministic counterparts are not the most efficient. An interesting results is that despite this limitation, the branch-and-cut algorithm provides orders of magnitude smaller computation times.

During the PhD thesis of Rémi Lucas, we worked on a robust railway scheduling problem in collaboration with the SNCF company [81]. The objective was not only to take into account the cost of the solution in the worst case scenario but also to obtain a solution that would only require a limited number of structural modifications once the uncertainty is revealed. Indeed, in many real-world applications, the cost of modifying an initially planned solution may be huge. Nevertheless, few robust approaches take this type of criterion into account. In Section 3.2 we generalize this approach in a new robust framework that minimizes the distance between the solutions before and after the uncertainty is revealed [P2]. We apply it to two flow problems for two distinct solution distances, and prove that all the corresponding robust problems are $\mathcal{NP}$-hard. We then identify a class of problems for which considering a discrete set of uncertain scenarios $\mathcal{U}$ is equivalent to considering its convex hull $conv(\mathcal{U})$. We prove that this result does not hold for three similar classes of problems. Finally, we present a case study on a railroad planning problem in which we compare our approach with two others from the literature.

1.2.3 Learning from the data

My last line of research, covered in Chapter 4, is dedicated to the combined use of operational research and machine learning techniques [P4, P9, P10, P13, P15, P18].

Section 4.1 is dedicated to a study conducted during the PhD thesis of Valentine Huré. We consider supervised classification which consists in learning a classifier able to predict as accurately as possible the class of labeled data. In order to produce a classifier that is both efficient and interpretable, we focus on the exact solution of the problem of building an optimal classification tree [P4]. We introduce three new MILPs for this problem and prove that they have a better relaxation than the most well-known formulation for this problem. Subsequently, we define a new iterative algorithm to fix the parameters of our models. Finally, we highlight the efficiency of our models and of our fitting algorithm through an extensive computational study. In particular, these experiments highlight that our approach is twice as fast as the exact state-of-the-art method, while providing similar or better predictions.

Section 4.2 focuses on learning B&B branching strategies in the context of repeated problems from the EDF company [P13]. We propose a reinforcement learning framework based on a Markov decision process for which we propose an original transition function that enables to take into account the structure of the B&B tree. We prove that when considering this transition function and a specific cost model, the problem of minimizing the size of the B&B tree can be solved optimally by a dynamic programming approach. Since the use of such algorithm is prohibitive in practice, we propose a Q-learning heuristic which leads in a few hours to satisfying results compared with the branching strategy of the commercial solver CPLEX. This work was undertaken during the PhD thesis of Marc Etheve [50] in collaboration with the EDF company.

Chapter 2

Improving the solution of hard MILPs

The MILP solution methods, such as branch-and-bound (B&B) and branch-and-cut (B&C), are fundamental techniques that involve exploring the solution space through the construction of an initially unknown-sized tree. To mitigate the exponential growth of the tree, solvers successfully employ various heuristic techniques. Nevertheless, as problem sizes increase, they generally rapidly become computationally intractable.

This chapter exemplifies the diverse approaches I have undertaken to extend the boundaries of MILP solving. Each of the two sections focuses on a specific problem, highlighting the theoretical and algorithmic contributions I have made in addressing this challenge.

2.1 A polyhedral approach for the K-partitioning problem

K-PARTITIONING PROBLEM
Input: A weighted graph $G = (V, E, w)$ and $K \in \mathbb{N}^*$.
Output: A partition of V in K clusters with minimal weight inside the clusters.

K-clustering is an $\mathcal{NP}$-hard supervised classification problem [57] for which many heuristics have been designed [82, 106]. To optimally solve graph partitioning problems, most of the linear programming approaches in the literature are based on *node-cluster* binary variables z_i^k taking value 1 if and only if node $i \in V$ is assigned to the cluster of index k. Unfortunately, these variables induce symmetry since permutations of the clusters indices k lead to equivalent solutions. Note that this can be more or less alleviated by additional constraints depending on the formulation considered [31, 51]. During my PhD, I introduced a symmetry-free formulation called *edge-representative formulation* (F_{er}) – detailed in Appendix A.1 – and studied its associated polyhedron [P7].

Let $v(P_1)$ be the optimal value of a problem (P_1) and let $(\overline{P_2})$ be the linear relaxation of a MILP (P_2). I present in this section the main results that I subsequently obtained on this topic in collaboration with Arnaud Knippel [P5, P6]:

- an extension (F_{ext}) of (F_{er}) with additional edge variables such that $v(\overline{F_{rep}}) \leqslant v(\overline{F_{ext}})$;
- the identification of a family of valid inequalities and the characterization of the conditions under which this family and constraints from (F_{ext}) are facet-defining;
- an efficient branch-and-cut algorithm that provides better numerical results than the direct resolution of (F_{er}), (F_{ext}) and two node-clusters formulations (F_{nc1}) and (F_{nc2});
- bounds on $v(\overline{F_{nc1}})$ and $v(\overline{F_{nc2}})$ which help to understand the poor numerical performances of these formulations.

2.1.1 Extended formulation

Our extended formulation (F_{ext}) considers the binary variables of (F_{er}): x_{ij} equals to 1 if and only if nodes i and j are in the same cluster, and r_i equals to 1 if and only if node i is the node of lowest index in its cluster. In that last case, node i is said to be the *representative* of its cluster. The formulation additionally includes a new set of binary edge variables $\tilde{x}_{ij}$ equal to 1 if and only if node j is represented by node i:

$$
(F_{ext}) \left\{
\begin{array}{lll}
\text{minimize} & \sum_{ij \in E} w_{ij} x_{ij} & \\
\text{subject to} & x_{ij} + x_{ik} - x_{jk} \leqslant 1 & i \in V\ j,\ k \in V \backslash \{i\},\ j < k \quad (2.1) \\
& r_j + \sum_{i=1}^{j-1} \tilde{x}_{ij} = 1 & j \in V \quad (2.2) \\
& \tilde{x}_{ij} \leqslant x_{ij} & ij \in E \quad (2.3) \\
& \tilde{x}_{ij} \leqslant r_i & ij \in E,\ i < j \quad (2.4) \\
& x_{ij} + r_i - \tilde{x}_{ij} \leqslant 1 & ij \in E,\ i < j \quad (2.5) \\
& \sum_{i=1}^{n} r_i = K & \quad (2.6) \\
& r_j + x_{ij} \leqslant 1 & i, j \in V,\ i < j \quad (2.7) \\
& r_i \in [0,1] & i \in V \\
& x_{ij},\ \tilde{x}_{ij} \in \{0,1\} & ij \in E
\end{array}
\right. .
$$

The triangle inequalities (2.1) ensure that for any distinct nodes i, j, and k, it is not possible to have exactly two of the edges (ij), (ik) and (jk) inside a cluster, thus guaranteeing the coherence of the partition. A node i is either a representative of its cluster or represented by a node $j < i$ through Constraints (2.2). The linearization Constraints (2.3), (2.4), and (2.5) ensure that $\tilde{x}_{ij}$ is equal to $r_i x_{ij}$. Exactly K clusters are obtained by Constraint (2.6). Constraints (2.7) are optional but enable to strengthen the formulation. Note that the integrity of variables r is ensured by that of variables $\tilde{x}$ through Constraints (2.2).

The only difference between (F_{er}) and (F_{ext}) is the constraints considered to ensure that each vertex is either a representative or is represented by a vertex with a lower index. Instead of Constraints (2.2) to (2.5), (F_{er}) considers

$$
r_j + \sum_{i=1}^{j-1} x_{ij} \geqslant 1 \qquad j \in V \tag{2.8}
$$

We now prove that the optimal value of the linear relaxation of (F_{er}) can not be better than that of (F_{ext}).

Theorem 2.1 *$v(\overline{F_{ext}}) \geqslant v(\overline{F_{er}})$ if $n \geqslant 4$ and $K \in \{2, \ldots, n-2\}$.*

Sketch of proof - We consider $proj(\overline{F_{ext}})$, the projection of $conv(\overline{F_{ext}})$ – the convex hull of the solutions of $\overline{F_{ext}}$ – onto the variable space of $\overline{F_{er}}$. We show that $proj(\overline{F_{ext}})$ is strictly included in $conv(\overline{F_{er}})$ by first identifying a point in $conv(\overline{F_{er}})$ that is not in $proj(\overline{F_{ext}})$ and then by proving that any point in $proj(\overline{F_{ext}})$ necessarily satisfies all the constraints of $\overline{F_{er}}$. □

This theorem shows that F_{ext} is stronger than F_{er} and that its resolution through branch-and-cut can thus lead to smaller search trees.

2.1.2 Facet defining inequalities

In order to define an efficient branch-and-cut algorithm, we first identify a new family of valid inequalities for (F_{ext}).

Property 2.2 *For each pair of nodes i and j with $i < j$, the sub-representative inequality*

$$x_{ij} \leqslant \sum_{h=1}^{i} \tilde{x}_{hj} \tag{2.9}$$

is valid for (F_{ext}).

Constraint (2.9) ensures that if nodes i and j are in the same cluster (i.e., $x_{ij} = 1$) then node j is represented by a node which index is at most i (i.e., $\sum_{h=1}^{i} \tilde{x}_{hj} = 1$).

Prior to determining if valid inequalities are facet-defining of $conv(F_{ext})$, we must determine its dimension.

Theorem 2.3 *The dimension of $conv(F_{ext})$ is equal to:*

(i) 0 *if* $K \in \{1, n\}$;

(ii) $|V|(|V|-2)+2$ *if* $K = 2$;

(iii) $|V|(|V|-2)$ *if* $K \in \{3, 4, \ldots, |V|-2\}$ *(i.e., it is full dimensional);*

(iv) $\frac{|V|(|V|-1)}{2} - 1$ *if* $K = |V|-1$.

Sketch of proof - The dimension is at most $|V|(|V|-2)$ since (F_{ext}) contains $|V|^2$ variables and since $2|V|$ of them can be substituted (using Constraints (2.2) and (2.6) and the fact that node 1 is always the representative of its cluster).

To prove that $conv(F_{ext})$ is full-dimensional, we assume that it is included in a hyperplane $H = \{x \in \mathbb{R}^{|V|(|V|-2)} \mid \alpha^T x = \alpha_0\}$ and prove that α is necessarily equal to the null vector. This can be done by considering several feasible solutions of the problem which lead to necessary conditions on the components of α. □

Since a polyhedral study of (F_{er}) has already been performed [P7], we focus on the new families of constraints. We consider the most general case $K \in \{3, 4, ..., |V|-2\}$ in which the polytope is full-dimensional.

Theorem 2.4 *The conditions under which constraints (2.3) - (2.5), and (2.9) are facet-defining of $conv(F_{ext})$ are summarized in Table 2.1.*

Sketch of proof - We prove that the face F induced by an inequality $a^T x \leqslant b$ is facet-defining by assuming, similarly to the previous proof, that F is included in a hyperplane $H = \{x \in \mathbb{R}^{|V|(|V|-2)} \mid \alpha^T x = \alpha_0\}$. We show that α is necessarily equal to λa with $\lambda \in \mathbb{R}$ by considering several solutions in F. □

Constraints	Conditions
(2.3)	if $i \geqslant 2$, $j \geqslant 4$
(2.4) and (2.5)	if $i \geqslant 4$
(2.9)	if and only if $i = 2$ or $i \geqslant 3$ and $K \leqslant \|V\| - 3$

Table 2.1. *Conditions under which family of constraints are facet-defining of $conv(F_{ext})$ when $K \in \{3, 4, ..., |V|-2\}$.*

We also obtain similar polyhedral results for the two variants of the problem in which the number of clusters is at least or at most K.

2.1.3 Branch-and-cut algorithm

Our branch-and-cut starts by a thorough cutting-plane step. In order to quickly solve the successive linear relaxations, we only keep in (F_{ext}) its smallest families of inequalities: Constraints (2.2) and (2.6). The constraints removed from the formulation are separated in this first step. We also separate sub-representative inequalities as well as 3 additional families of valid inequalities that have previously been identified, namely: the 2-partition inequalities, the general clique inequalities and the paw inequalities. For each family, we define a fast greedy algorithm and a slower local search algorithm inspired by the work of Kernighan and Lin [74]. The slower algorithms are only used when no violated inequality is found by the greedy algorithms in order to extend the cutting plane step. At each step, at most 3000 violated inequalities are separated and the 500 most violated are added to the model.

Once no more violated inequality is found for a continuous solution x^c, we consider (F_{ext}) with all its constraints plus all the previously generated inequalities that are *tight* for x^c (i.e., inequalities $a^T x \leqslant b$ such that $a^T x^c = b$). A classical branch-and-cut is then performed on this formulation in which the slower separation algorithms are not considered.

We compare the performances of this algorithm with the direct resolution through CPLEX of (F_{er}), (F_{ext}) as well as of two node-cluster partitioning formulations (F_{nc1}) and (F_{nc2}) that we adapted to the K-clustering problem and that are detailed in Appendix A.2. Table 2.2 illustrates some of the results obtained when clustering cities from instances of the traveling salesman problem. When the time limit of one hour is reached, the relative gap between the best upper and lower bounds on the optimal solution is represented. We can see that the node-cluster formulations have the worst performances and that the branch-and-cut has the best. Moreover, we can observe on instances solved optimally that (F_{ext}) leads to smaller search trees than (F_{er}). This is probably due to its stronger linear relaxation.

Instance	**n**	**K**	**Time (s) and Gap (%)**										**Nodes**				
			(F_{nc1})		(F_{nc2})		(F_{er})		(F_{ext})		(BC)		(F_{nc1})	(F_{nc2})	(F_{er})	(F_{ext})	(BC)
			time	gap	time	gap	time	gap	time	gap	time	gap					
bayg	**29**	**2**	19	**0**	468	**0**	3251	**0**	112	**0**	**3**	**0**	1534	4287	35141	265	0
		4	TL	10	TL	19	TL	1	TL	1	**156**	**0**	210026	22604	41339	34090	57
		6	TL	16	TL	9	230	**0**	304	**0**	**14**	**0**	230472	42730	6442	4875	0
		8	TL	16	1966	**0**	40	**0**	109	**0**	**9**	**0**	215260	33339	1541	1859	0
att	**48**	**2**	**26**	**0**	1231	**0**	TL	5	1388	**0**	35	**0**	167	140	1385	197	0
		4	552	**0**	3503	**0**	TL	4	2285	**0**	**18**	**0**	5035	2596	4142	1940	0
		6	TL	9	TL	6	TL	7	TL	6	**19**	**0**	33152	2645	19753	5477	0
		8	TL	13	TL	2	TL	5	TL	3	**14**	**0**	25983	4174	46173	10428	0
st	**70**	**2**	629	**0**	TL	12	197	**0**	308	**0**	**86**	**0**	859	0	0	0	0
		4	TL	4	TL	7	TL	3	TL	2	**TL**	**0**	2941	31	79	49	0
		6	TL	13	TL	10	TL	5	TL	4	**566**	**0**	4342	142	694	206	5
		8	TL	16	TL	11	TL	8	TL	7	**TL**	**3**	3557	319	1300	350	8
eil	**101**	**2**	TL	9	TL	19	TL	4	TL	4	**TL**	**1**	1287	0	0	0	0
		4	TL	19	TL	20	TL	10	TL	9	**TL**	**6**	294	0	3	0	0
		6	TL	35	TL	32	TL	24	TL	24	**TL**	**18**	57	0	7	2	0
		8	TL	34	TL	30	TL	24	TL	23	**TL**	**16**	41	0	39	5	0
bier	**127**	**2**	TL	25	TL	37	TL	32	TL	31	**TL**	**15**	869	0	0	0	0
		4	TL	54	TL	54	TL	49	TL	48	**TL**	**31**	10	0	2	0	0
		6	TL	69	TL	69	TL	65	TL	64	**TL**	**53**	13	0	2	0	0
		8	TL	65	TL	63	TL	59	TL	58	**TL**	**47**	7	0	0	0	0

Table 2.2. *Average results (in terms of time in seconds, gap and number of nodes in the branch-and-cut tree) obtained for each of the five K-partitioning methods over 5 instances of the TSPLIB [95]. (BC) corresponds to the branch-and-cut algorithm. Bold results are the best in terms of gap and time. A gap of 0% corresponds to an optimal solution. The symbol TL indicates that the time limit of one hour is reached.*

2.1.4 Bounds on $v(\overline{F}_{nc1})$ and $v(\overline{F}_{nc2})$

To further understand these results, we determine bounds that prove that the optimal value of the linear relaxations of the two node-cluster formulations are low.

Property 2.5

- $v(\overline{F_{nc2}}) \leqslant \min_{j\in\{2,\dots,K\}} \min_{i<j} \frac{w_{ij}}{2^{K-j+1}}$.
- *If* $K \in \{3,\dots,n\}$, *then* $v(\overline{F_{nc1}}) \leqslant \min_{j\in\{2,\dots,K-1\}} \min_{i<j} \frac{w_{ij}}{2^{K-j}}$.
- *If* $K = 2$, *then* $v(\overline{F_{nc1}}) \in [\min_{i,j\in V} w_{ij}\frac{n-1}{2}, \frac{1}{2}\sum_{i=2}^{n} w_{1i}]$.

Sketch of proof - The upper bounds are obtained by building feasible solutions of the linear relaxations and taking the minimal value of their objective function. For the lower bound, we deduce from the constraints of (F_{nc1}) that, when $K = 2$, $\sum_{ij\in E} x_{ij}$ is necessarily greater than or equal to $\frac{n-1}{2}$. □

The fact that $v(\overline{F_{nc1}})$ is higher when $K = 2$ is coherent with the numerical results.

We now consider the p-median problem that we efficiently solve through a Benders decomposition which enables to significantly reduce the number of constraints considered in the problem.

2.2 A Benders decomposition for the p-median problem

p-MEDIAN PROBLEM	
Input:	A set of clients V, a set of sites U, distances $\{d_{ij}\}_{(i,j)\in V\times U}$ and $p \in \mathbb{N}^*$.
Output:	p sites to open such that the sum of the distances between each client and its closest opened site is minimized.

The p-median is an $\mathcal{NP}$-hard discrete location problem [71]. It has many applications in which the sites can for example be warehouses, factories, shelters, public services, etc. When the set of clients and sites are identical, it corresponds to a clustering problem known as the *k-medoids problem* [73]. Similarly to the K-clustering problem, various heuristics and meta-heuristics have been proposed in the literature for this problem [14, 70, 86, 94] but the exact resolution of large instances remains a challenge.

The results presented in this section have been obtained during the PhD of Cristian Duran which I co-supervise with Sourour Elloumi [P11]. The main contributions are:

- a Benders decomposition of the p-median problem;
- an efficient algorithm to separate the Benders cuts;
- a two-step Benders decomposition algorithm. In the first step, the integrity of the variables is relaxed which enables to obtain quickly very strong bounds on the optimal solution;
- an extensive computational study which highlights that our algorithm provides better results than state of the art methods.

2.2.1 Benders decomposition

Benders decomposition [19] is a technique that splits the optimization problem into a *master problem* (MP) and one or several *sub-problems* (SP). This method takes advantage of the fact that fixing the integer variables of a MILP yields a linear program whose dual can be solved to obtain valid inequalities known as *Benders cuts*. The (MP) contains the integer variables, and its solution enables to fix their values in the sub-problems. These problems are solved iteratively until no violated Benders cut is found.

All the MILP formulations of the p-median problem associate a binary variable y_j to each site $j \in U$ equal to 1 if and only if site j is opened. Our Benders decomposition is based on a formulation (F) that additionally considers one variable for each client $i \in V$ and each distinct distance between i and a site [49]. Let K_i be the number of different distances from i to any site and let $D_i^1 < D_i^2 < ... < D_i^{Ki}$ be these distances sorted. Let the *allocation distance* of client $i \in V$ be the distance between i and its closest opened site. Binary variable z_i^k is equal to one if and only if the allocation distance of client $i \in V$ is at least D_i^{k+1}:

$$
(F) \begin{cases}
\text{minimize} & \sum_{i \in V} \left(D_i^1 + \sum_{k=1}^{K_i - 1} (D_i^{k+1} - D_i^k) z_i^k \right) & & \\
\text{subject to} & \sum_{j \in U} y_j = p & & (2.10) \\
& z_i^1 + \sum_{j \in U : d_{ij} = D_i^1} y_j \geqslant 1 & i \in V & (2.11) \\
& z_i^k + \sum_{j \in U : d_{ij} = D_i^k} y_j \geqslant z_i^{k-1} & i \in V,\ k \in \{2, ..., K_i\} & (2.12) \\
& z_i^k \geqslant 0 & i \in V,\ k \in \{1, ..., K_i\} & \\
& y_j \in \{0, 1\} & j \in U &
\end{cases}
$$

Constraint (2.10) ensures that exactly p sites are opened. Constraints (2.11) and (2.12) impose that z_i^k is equal to 1 if there is no site opened at a distance lower than D_i^{k+1} of client i.

Note that only the y variables are required to be integer. Consequently, our Benders decomposition is based on an (MP) including the y variables as well as a new set of continuous variables θ_i representing the allocation distance of each client $i \in V$:

$$
(MP) \begin{cases}
\text{minimize} & \sum_{i \in V} \theta_i & & \\
\text{subject to} & \sum_{j \in U} y_j = p & & \\
& \theta_i \text{ satisfies } BD_i & i \in V & \\
& y_j \in \{0, 1\} & j \in U &
\end{cases}
$$

where BD_i is the set of benders cuts generated for client i. These sets are initially empty and grow through the iterations to ensure that the allocation distances of the clients $\{\theta_i\}_{i \in V}$ are not underestimated.

The sub-problem can be decomposed into $|V|$ sub-problems, each one computing the allocation distance of a single client. Given a feasible solution $\overline{y}$ of (MP) or of its linear relaxation $(\overline{MP})$, the sub-problem of client i is defined by:

$$
SP_i(\overline{y}) \begin{cases}
\text{minimize} & D_i^1 + \sum_{k=1}^{K_i - 1} (D_i^{k+1} - D_i^k) z_i^k & & \\
\text{subject to} & z_i^1 \geqslant 1 - \sum_{j \in U : d_{ij} = D_i^1} \overline{y}_j & & \\
& z_i^k - z_i^{k-1} \geqslant 1 - \sum_{j \in U : d_{ij} = D_i^k} \overline{y}_j & k \in \{2, ..., K_i\} & \\
& z_i^k \geqslant 0 & k \in \{1, 2, ..., K_i\} &
\end{cases}
$$

The objective function corresponds to the allocation distance of client i while the constraints are similar to Constraints (2.11) and (2.12).

We now determine the expression of the unique Benders cut that can be obtained from this problem.

Theorem 2.6 *Let $\tilde{k}_i$ be the largest index in $\{1, 2, ..., K_i\}$ which satisfies $\sum_{j \in U\,:\, d_{ij} \leqslant D_i^k} \overline{y}_j < 1$ for a given solution $\overline{y}$ of (MP) or $(\overline{MP})$. The unique Benders cut obtained from $SP_i(\overline{y})$ is $\theta_i \geqslant D_i^1$ if $\tilde{k}_i = 0$ and $\theta_i \geqslant D_i^{\tilde{k}_i+1} - \sum_{j \in U\,:\, d_{ij} \leqslant D_i^{\tilde{k}_i}} (D_i^{\tilde{k}_i+1} - d_{ij}) y_j$ otherwise.*

Sketch of proof - We first determine that $SP_i(\overline{y})$ necessarily has one unique solution whose expression we determine. We deduce from it the optimal solution of the dual of $SP_i(\overline{y})$ through the complementary slackness conditions which directly leads to the expression of the Benders cut. □

We observe that these inequalities are the same as those obtained in previous decompositions from another (pMP) formulation [37, 83]. This was quite unexpected since, even though the master problems are the same in the two decompositions, the sub-problems are different.

We consider the polytope associated with (MP) when for all clients $i \in V$ the set BD_i contains the Benders cuts associated with all possible indexes $\tilde{k}_i \in \{0, 1, ..., K_i - 1\}$:

$$(MP_F) = \left(\begin{array}{l|ll} y \in \{0,1\}^M & \sum_{j=1}^{M} y_j = p & (2.13) \\ \theta \in \mathbb{R}^{N+} & \theta_i \geqslant D_i^k - \sum_{j \in U \,|\, d_{ij} \leqslant D_i^k} (D_i^k - d_{ij}) y_j \quad i \in \mathcal{N}\ k \in \{1, ..., K_i\} & (2.14)_{i,k} \end{array} \right)$$

We now determine when the Benders cuts are facet-defining of $conv(MP_F)$, the convex hull of the solutions of (MP_F).

Theorem 2.7 *The dimension of $conv(MP_F)$ is $|U| + |V| - 1$ if $p < |U|$ and $K_i > 1$ for all clients $i \in V$. Otherwise, the dimension is lower.*

For a given client $\underline{i} \in V$ and index $\underline{k} \in \{1, 2, ..., K_{\underline{i}}\}$, let $F_{\underline{i},\underline{k}}$ be the face of $conv(MP_F)$ defined by inequality $(2.14)_{\underline{i},\underline{k}}$. We focus on the case in which the dimension of $conv(MP_F)$ is maximal (i.e., $p < |U|$ and $K_i > 1$ for all clients $i \in V$) and characterize the conditions under which $F_{\underline{i},\underline{k}}$ is facet-defining.

Theorem 2.8 *If $p < |U|$ and $K_i > 1$ for all clients $i \in V$, then $F_{\underline{i},\underline{k}}$ defines a facet of $conv(MP_F)$ if and only if $|\{j \in U \,|\, d_{ij} \geqslant D_{\underline{i}}^{\underline{k}}\}| \geqslant p \geqslant 2$.*

Sketch of proof - The techniques used in the two previous proofs are similar to the ones used for the K-clustering problem in Section 2.1. The additional difficulty here resides in the identification of relevant feasible solutions in $F_{\underline{i},\underline{k}}$. □

Thus, when the dimension of $conv(MP_F)$ is maximal, most of the Benders cuts are facet-defining. This highlights the relevance of a solution algorithm based on their generation.

2.2.2 Solution algorithm

Thanks to Theorem (2.6) we know that generating the Benders cut of client $i \in V$ only requires to compute index $\tilde{k}_i$. We show that this can be done in $\mathcal{O}(|U|)$ by computing a priori for each client $i \in V$ and each index $r \in \{1, 2, ..., |U|\}$, the r^{th} closest site of client i. This algorithm is detailed in Appendix B.

We use this separation algorithm in a two-step decomposition algorithm. A strong initial upper bound is obtained through the solution provided by Popstar [96] which, to the best of our knowledge, is the best heuristic for the (pMP). To quickly generate Benders cuts, the integrity of the y variables of (MP) are relaxed in the first step. At each iteration, a feasible solution is generated from the fractional solution of $(\overline{MP})$ by opening the p sites for which the value of the y variables are maximal.

At the end of the first phase, in order to reduce the size of the MILP, we do not keep all the generated Benders cuts. Instead, for each client $i \in V$, we identify the largest index $\tilde{k}_i$ of a generated cut that is tight for the last fractional solution obtained and ignore all the cuts obtained from a larger index. Moreover, we apply a reduced cost fixing step to fix the value of y variables to 0 or 1 whenever possible.

In the second step, the integrity constraints on the y variables are added and the problem is solved through a branch-and-cut. To ensure the validity of each integer solution found, the associated sub-problems are solved and any violated Benders cut is added.

An extensive computational study of more than 250 instances from 6 distinct datasets enables us to establish that this algorithm is extremely efficient compared with many approaches from the literature. It is significantly faster than state-of-the-art methods including heuristics. Table 2.3 presents a representative extract of our results obtained on TSP instances. Column "Objective" contains the objective value of the solution returned by our method. This value is in bold if it is the first time that the corresponding instance is solved to optimality or if the best known solution for this instance is improved. Columns "LB" and "UB" represent the bounds on the optimal objective value obtained at the end of the first phase of our algorithm. Column "Iter." corresponds to the number of solutions for which Benders cuts were separated. The two last columns are dedicated to Zebra [56], the best exact solution method in the literature that we run on the same computer than our Benders decomposition. We are able to solve to optimality 7 more instances than Zebra and when both methods are optimal, we are significantly faster.

Instance				Phase 1			Phase 1 + 2			Zebra	
Name	$\lvert U\rvert = \lvert V\rvert$	p	**Objective**	**LB**	**UB**	**Time**	**Gap**	**Iter.**	**Time**	**Gap**	**Time**
usa13509	13509	10	**398561730**	398561600	398561730	288	0%	10	755	∞	⧫
usa13509	13509	100	**108002205**	107983102	108002411	523	0%	23	4043	∞	⧫
usa13509	13509	1000	29268216	29262339	29309009	154	0%	31	1382	0%	TL
usa13509	13509	2000	18230856	18229432	18238229	47	0%	21	125	0%	584
usa13509	13509	3000	13098935	13097929	13101469	49	0%	28	72	0%	1674
usa13509	13509	4000	9905715	9905071	9910848	37	0%	17	50	0%	166
usa13509	13509	5000	7608605	7608242	7611958	45	0%	22	61	0%	86
sw24978	24978	10	**22670073**	22670073	22670073	1037	0%	12	1037	∞	⧫
sw24978	24978	100	**6660424**	6657806	6660424	9634	0,031%	14	TL	∞	⧫
sw24978	24978	1000	**1841723**	1841613	1844801	621	0%	23	3814	0%	30796
sw24978	24978	2000	**1197278**	1197231	1198464	208	0%	17	476	0%	TL
sw24978	24978	3000	**911361**	911308	911988	145	0%	17	344	0%	3614
sw24978	24978	4000	**737645**	737602	738045	92	0%	15	190	0%	TL
sw24978	24978	5000	**617637**	617593	618096	76	0%	18	127	0%	TL

Table 2.3. *Results on large TSP instances for our method and Zebra. TL indicates that the time limit of 36000 seconds has been reached. The symbol ⧫ means that the computer ran out of memory. A bold objective value indicates that it is the first time the instance is solved to optimality or that we improve the best-known value.*

Chapter 3

Dealing with uncertainty

In this chapter we now assume that the exact value of some coefficients of the MILP are not known exactly and focus on robust optimization to handle this uncertainty. Following the work of Soyster [98] robust approaches often search a solution which is feasible for any value of the data in an uncertainty set [18, 27]. However, in many concrete problems, a competitive solution may not be feasible for all possible realizations of the uncertainty. To alleviate this problem, two-stage robust optimization consider recourse variables which value is computed only once the uncertainty is revealed [8, 17]. In this chapter, we present two robust two-stage approaches.

We first focus in Section 3.1 on a robust weighted vertex p-center problem. We consider a standard robust framework in which the objective function of the deterministic problem is optimized.

Then, in Section 3.2, we motivate and introduce a new generic robust framework in which the objective is now to minimize the distance between the solutions before and after the uncertainty is revealed.

3.1 A two-stage robust weighted vertex p-center problem

WEIGHTED VERTEX p-CENTER PROBLEM	
Input:	A set of clients V, a set of sites U, weights $\{w_i\}_{i\in V}$, distances $\{d_{ij}\}_{(i,j)\in V\times U}$, and $p \in \mathbb{N}^*$.
Output:	Select p sites to open such that the maximal weighted distance between a client and its closest opened site is minimized.

The weighted vertex p-center problem (PCP) is a generalization of the $\mathcal{NP}$-hard p-center problem which corresponds to the case where all the clients weights $\{w_i\}_{i\in V}$ are equal to 1. The only difference between the p-center problem and the p-median problem considered in Section 2.2 is the objective function. In the former it is equal to the maximal allocation distance of a client and in the latter it corresponds to the sum of all clients allocations.

The incorporation of uncertainty in (PCP) has important applications in emergency logistics problems, where a prompt response is required following a disaster (such as earthquakes, tsunami, or landslides). The consequences of these disasters make it challenging to precisely estimate the demand for relief materials or the travel times between relief centers and affected locations [97]. In robust versions of this problem, two approaches can be considered depending on whether the clients allocation to the opened sites are made before [9, 80] or after [40, 47] the uncertainty is revealed. The first approach leads to *single-stage problems* while the second leads to more difficult *two-stage problems* in which the clients allocation are recourse variables. In both approaches, the choice of the opened sites are made before the uncertainty is revealed since the construction of a relief center requires time. Most of the works in this context have been focused on the investigation of integer programming modeling and heuristic resolution approaches [13, 64, 90, 102, 103].

The work presented in this section has been initiated during the master thesis of Natalia Bocanegra which was also supervised by Cristian Duran and Sourour Elloumi [P12]. Our main contributions are:

- the definition of a robust two-stage variant of (PCP) called $(RPCP)$;
- the definition of robust reformulations of $(RPCP)$ based on 5 different MILP formulation of (PCP);
- the proof that a finite subset of the uncertainty set can be considered without losing optimality;
- the definition of two exact algorithms based on this result: a column-and-constraint generation algorithm and a branch-and-cut algorithm;
- the evaluation of their efficiency on randomly generated instances.

3.1.1 Problem definition

In robust optimization, the uncertainty is generally represented by coefficients which can take any value in an uncertainty set. Each realization of the uncertainty set is called a *scenario*. The most classical uncertainty sets are the box, the ellipsoidal and the budgeted uncertainty sets [16, 27, 46, 91].

Several robust variants of (PCP) with either a single-stage or two-stages have been considered. For example, [9] consider uncertain clients weights, minimize the regret of the worst-case scenario and show that the problem can be solved through a number of particular (PCP). In [10] a box uncertainty set for both the clients weights and the edge lengths is considered for $p = 1$. A polynomial algorithm to find the robust solution for the problem on a tree is presented. In [80] a similar uncertainty set is considered for the single-stage robust (PCP) and a simulating annealing heuristic is developed to solve it. Du and Zhou propose a two-stage robust model for a reliable facility location problem in which the clients of disrupted facilities can be reallocated [47]. They propose three solution methods: a linear reformulation, a Benders dual cutting planes method, and a *column-and-constraint generation* method. Demange et al. introduce the robust p-center problem under pressure motivated by the context of locating shelters for evacuation in case of wildfires, where the uncertainty is in the available network connections [40]. They present a MILP formulation and a decomposition scheme to solve it.

Following [80], we consider uncertain weights and distances that can take any value in a box uncertainty set. More precisely, the weight w_i of client $i \in V$ is assumed to be in $[w_i^-, w_i^+]$ with $0 \leqslant w_i^- \leqslant w_i^+$, while the distance d_{ij} between client $i \in V$ and site $j \in U$ takes its value in $[d_{ij}^-, d_{ij}^+]$ with $0 \leqslant d_{ij}^- \leqslant d_{ij}^+$. The box uncertainty set we consider $\Omega \subset \mathbb{R}^{|V|+|U|\times|V|}$ is the Cartesian product of intervals $[w_i^-, w_i^+]$ and $[d_{ij}^-, d_{ij}^+]$ for each $i \in V$ and $j \in U$.

Before formally defining problem $(RPCP)$, we first introduce several notations. Let $Y = \{y \in \{0,1\}^{|U|} \mid \sum_{j\in U} y_j = p\}$ be the set of vectors representing p opened sites and let $J_y = \{j \in U \mid y_j = 1\}$ be the set of opened sites for vector $y \in Y$. Let w_i^ω and d_{ij}^ω respectively be the weight of client $i \in V$ and the distance between client i and site $j \in U$ in a given scenario $\omega \in \Omega$.

In $(RPCP)$, the clients allocations are determined after both the opening of the sites and the revelation of the uncertainty. Consequently, we define the maximal weighted distance between a client and its closest opened site, also called *radius*, for a given set of opened sites $y \in Y$ and a given scenario $\omega \in \Omega$:

$$R(\omega, y) = \max_{i\in V} \min_{j\in J_y} w_i^\omega d_{ij}^\omega \tag{3.1}$$

This corresponds to the optimal value of the deterministic (PCP) when sites J_y are opened and when the uncertain coefficients take the value ω. Let $y^*(\omega) \in Y$ be a vector that minimizes the radius when scenario ω occurs and let R_ω^* be the value of this minimal radius (i.e., $R_\omega^* = R(\omega, y^*(\omega)) = \min_{y\in Y} R(\omega, y)$).

Problem $(RPCP)$ consists in finding a solution $y \in Y$ that minimizes the regret of the worst-case scenario:

$$(RPCP): \min_{y \in Y} \max_{\omega \in \Omega} R(\omega, y) - R^*_\omega \tag{3.2}$$

3.1.2 MILP formulations

To model $(RPCP)$, we adapt five (PCP) formulations from the literature. All these deterministic formulations contain a set of variables $\{y_j\}_{j \in U}$ equal to 1 if and only if site j is opened. As summarized in Table 3.1, these formulations consider various sets of additional variables in order to represent the value of the radius:

- $x_{ij} \in \{0,1\}$ for $(i,j) \in V \times U$ is equal to 1 if and only if client i is allocated to site j;
- $z^k \in \{0,1\}$ for $k \in \{1,2,...,K\}$ is similar to variable z^k_i of the p-median formulation (F) in Section 2.2 and is equal to 1 if and only if the radius is greater than D^k with $D^0 < D^1 < ... < D^K$ all the distinct weighted distances between clients and sites;
- $u_k \in \{0,1\}$ for $k \in \{1,...,K\}$ is equal to 1 if and only if the radius is equal to D^k;
- $r \in [D^0, D^K]$ is equal to the radius.

Formulation		**Variables**				
		x_{ij}	y_j	z^k	u^k	r
(F_1)	[39]	x	x			x
(F_2)	[P1]		x	x		
(F_3)	[P1]		x			x
(F_4)	[65]		x		x	
(F_5)	[55]		x			x

Table 3.1. *Sets of variables considered in the different formulations of the p-center problem considered.*

Note that Formulations (F_2) and (F_3) were introduced by Sourour Elloumi and I in a previous work [P1].

To model the robust problem $(RPCP)$ variables x_{ij}, u^k_i, and z^k_i must be duplicated for each possible scenario since the clients allocations and the radius can change from one scenario to another. As formulations (F_3) and (F_5) do not consider these variables, their adaptation to $(RPCP)$ are the only ones in which the number of variables is not proportional to the number of scenarios. We will see in Section 3.1.4 that this feature will make possible the use of a branch-and-cut algorithm for these two formulations that will significantly outperform the alternative solution method. Since the adaptation of the formulations to the robust case are similar and since the number of pages in this thesis is limited, we focus on one formulation in which the number of variables depends on the number of scenarios (F_1) and one in which it does not (F_3) to highlight their differences. The definition of the three other formulations and their robust counterparts are detailed in Appendix C. Formulations (F_1) and (F_3) are defined as follows:

$$
(F_1)\begin{cases}
\min & r & & \\
\text{s.t.} & r \geqslant \displaystyle\sum_{j\in U} w_i d_{ij} x_{ij} & i \in V & (3.3)\\
& \displaystyle\sum_{j\in U} x_{ij} = 1 & i \in V & (3.4)\\
& x_{ij} \leqslant y_j & i \in V\ j \in U & (3.5)\\
& \displaystyle\sum_{j\in U} y_j = p & & (3.6)\\
& y_j \in \{0,1\} & j \in U & \\
& x_{ij} \in \{0,1\} & i \in V\ j \in U &
\end{cases}
$$

$$
(F_3)\begin{cases}
\min & r & & \\
\text{s.t.} & r + D^k \displaystyle\sum_{j\in U\,:\,w_i d_{ij} < D^k} y_j \geqslant D^k & i \in V\ k \in \{1,...,K\} & \\
& & \text{s.t. } \exists j \in U\ d_i t_{ij} = D^k & (3.7)\\
& \displaystyle\sum_{j\in U} y_j = p & & (3.8)\\
& y_j \in \{0,1\} & j \in U &
\end{cases}
$$

Constraints (3.3) ensure that the radius is not lower than the weighted distance between a client and its allocated site. Similarly, Constraints (3.7) imposes that the radius is at least D^k if there is no opened site at distance less than D^k of client i. Each client is assigned to exactly one site through Constraints (3.4) and a site can not be assigned to a client if it is not opened via Constraints (3.5). Finally, Constraints (3.6) and (3.8) set the number of opened sites to p.

Our robust formulation based on (F_1) considers one set of binary allocation variables $\{x_{ij}^{\omega}\}_{(i,j)\in V\times U}$ for each scenario $\omega \in \Omega$ such that x_{ij}^{ω} is equal to 1 if and only if client i is assigned to site j when scenario ω occurs:

$$
(RF_1)\begin{cases}
\text{minimize} & v & & \\
\text{subject to} & v \geqslant \displaystyle\sum_{j\in U} w_i^{\omega} d_{ij}^{\omega} x_{ij}^{\omega} - R_{\omega}^{*} & i \in V\ \omega \in \Omega & (3.9)\\
& \displaystyle\sum_{j\in U} x_{ij}^{\omega} = 1 & i \in V\ \omega \in \Omega & (3.10)\\
& x_{ij}^{\omega} \leqslant y_j & i \in V\ \ j \in U\ \omega \in \Omega & (3.11)\\
& \displaystyle\sum_{j\in U} y_j = p & & \\
& y_j \in \{0,1\} & j \in U & \\
& x_{ij}^{\omega} \in \{0,1\} & i \in V\ j \in U\ \omega \in \Omega &
\end{cases}
$$

Constraints (3.9), (3.10) and (3.11) correspond to Constraints (3.3), (3.4) and (3.5) but applied to each scenario $\omega \in \Omega$. In Constraints (3.9), R_{ω}^{*} is additionally subtracted so that v is at least equal to the regret of each scenario $\omega \in \Omega$.

We now present the adaption of formulation (F_3) to $(RPCP)$:

$$(RF_3)\begin{cases} \text{minimize} & v & & \\ \text{subject to} & v \geqslant D^k_\omega(1 - \sum\limits_{j\in U\,:\,w^\omega_i d^\omega_{ij} < D^k_\omega} y_j) - R^*_\omega & \omega \in \Omega\ i \in V & \\ & & k \in \{1,2,...,K_\omega\} \mid \exists j \in U\ w^\omega_i d^\omega_{ij} = D^k_\omega & \\ & \sum\limits_{j\in U} y_j = p & & \\ & y_j \in \{0,1\} & j \in U & \end{cases} \tag{3.12}$$

with K_ω the number of distinct values $\{w^\omega_i d^\omega_{ij}\}_{i\in V, j\in U}$ in scenario $\omega \in \Omega$ and D^k_ω the k^{th} smallest of these values.

Since Ω contains an infinite number of scenarios, the number of constraints of (RF_1) and (RF_3) is also infinite. Additionally, (RF_1) also has an infinite number of variables. We show in the next section how this problem can be circumvented to enable the practical resolution of $(RPCP)$.

3.1.3 Reduction of the number of scenarios

In this section we prove that $(RPCP)$ can be solved by considering at most $|V|$ scenarios for each solution $y \in Y$. We first define these scenarios.

Definition 3.1 *Let $y \in Y$ be a feasible solution and let $\underline{i} \in V$ be any client. We define $\omega_{\underline{i}}(y)$ as the following scenario:*

- $w_i^{\omega_{\underline{i}}(y)} = \begin{cases} w_i^+ & \text{if } i = \underline{i} \\ w_i^- & \text{otherwise} \end{cases}$
- $d_{ij}^{\omega_{\underline{i}}(y)} = \begin{cases} d_{ij}^+ & \text{if } i = \underline{i} \text{ and } y_j = 1 \\ d_{ij}^- & \text{otherwise} \end{cases}$

We now prove that at least one of the scenarios in $\{\omega_i(y)\}_{i\in V}$ leads to a maximal regret for $y \in Y$.

Theorem 3.1 *Let $y \in Y$ be a first-stage solution of $(RPCP)$. There exists $i \in V$ such that $\omega_i(y)$ is an optimal solution of $\max_{\omega\in\Omega} R(\omega, y) - R^*_\omega$.*

Sketch of proof - We consider a scenario $\overline{\omega} \in \Omega$ that leads to a maximal regret for solution y (i.e., $\overline{\omega} = argmax_{\omega\in\Omega} R(\omega, y) - R^*_\omega$). We iteratively modify this scenario so that it becomes equal to one scenario in $\{\omega_i(y)\}_{i\in V}$ and ensure at each iteration that the regret is not reduced. □

Theorem 3.1 enables to only consider a finite set of scenarios without losing the optimality:

$$(RPCP): \min_{y\in Y} \max_{\omega\in\{\omega_i(y)\}_{i\in V}} R(\omega, y) - R^*_\omega \tag{3.13}$$

Nevertheless, the direct resolution of $(RPCP)$ remains a challenge as the number of scenarios to consider is equal to $|Y| \times |V|$ which is exponential in the number of sites.

3.1.4 Solution algorithms

Column-and-constraint generation algorithm

To solve $(RPCP)$, we first define a column and constraint generation algorithm $(C\&CG)$ that takes as an input any of our robust formulations denoted by (RF). In this algorithm, a set of scenarios $\overline{\Omega}$ initially empty is considered. A solution (y, v) is obtained by solving optimally a relaxation of (RF) in which we only consider scenarios $\overline{\Omega}$. We then compute the regret of each scenario $\{\omega_i(y)\}_{i\in V}$ and if any is greater than v, the scenario with the largest regret is added to $\overline{\Omega}$. This process is repeated until no violated scenario is found. The detailed algorithm is presented in Appendix C.3.

Branch-and-cut algorithm

The main drawback of algorithm $(C\&CG)$ is that the resolution of (RF) is restarted from scratch each time a scenario is added to $\overline{\Omega}$. To improve the performances, we propose a branch-and-cut algorithm that only solves (RF) once. During this resolution, each feasible solution (y, v) found by the solver is accepted only if it satisfies all the scenarios in $\{\omega_i(y)\}_{i\in V}$. Otherwise, the scenarios with the largest regret is added to $\overline{\Omega}$. This process can be performed through callbacks which are features of MILP solvers.

For formulations (RF_1), (RF_2) and (RF_4), the addition of a scenario to $\overline{\Omega}$ requires the addition of new variables to the model which is not possible in a callback. Consequently, this approach is only possible for formulations (RF_3) and (RF_5).

Reducing the number of deterministic problems solved

Each time a feasible solution (y, v) of (RF) is obtained, our two algorithms must determine if this solution satisfies scenario $\omega_i(y)$ for each $i \in V$ (i.e., if $R(\omega_i(y), y) - R^*_{\omega_i(y)} \leqslant v$). Computing $R(\omega_i(y), y)$ only requires for each client to find its closest center in J_y which can be done in polynomial time. However, computing $R^*_{\omega_i(y)}$ requires the resolution of a deterministic p-center problem which is $\mathcal{NP}$-hard. This resolution can be avoided if we know beforehand that (y, v) satisfies scenario $\omega_i(y)$. This is the case if there exists a lower bound R^{*i}_{lb} on $R^*_{\omega_i(y)}$ such that $R(\omega_i(y), y) - R^{*i}_{lb} \leqslant v$.

To obtain such lower bounds, we consider the following scenarios.

Definition 3.2 *Let $\underline{i} \in V$. We define $\omega^{\underline{i}}_{lb}$ such that for all $i \in V$ and all $j \in U$, $d_{ij}^{\omega^{\underline{i}}_{lb}} = d^-_{ij}$, and*

$$w_i^{\omega^{\underline{i}}_{lb}} = \begin{cases} w_i^+ & \text{if } i = \underline{i} \\ w_i^- & \text{otherwise} \end{cases}.$$

The following lemma shows that the optimal value of the deterministic PCP associated with scenario ω^i_{lb} provides a lower bound on $R^*_{\omega_i(y)}$.

Lemma 3.1 *For any $i \in V$ and $y \in Y$, $R^*_{\omega^i_{lb}} \leqslant R^*_{\omega_i(y)}$.*

Proof - By definition of $R^*_{\omega^i_{lb}}$, we know that $R^*_{\omega^i_{lb}} \leqslant R(\omega^i_{lb}, y^*(\omega_i(y)))$. Moreover, since the only difference between scenarios $\omega_i(y)$ and ω^i_{lb} is a reduction of the distances $R(\omega^i_{lb}, y^*(\omega_i(y))) \leqslant R^*_{\omega_i(y)}$. □

Note that unlike $\{\omega_i(y)\}_{i\in V}$, scenarios $\{\omega^i_{lb}\}_{i\in V}$ do not depend on any feasible solution $y \in \Omega$. Therefore, the $|V|$ corresponding lower bounds $\{R^*_{\omega^i_{lb}}\}_{i\in V}$ can all be computed in a pre-processing step to reduce the number of deterministic p-center solved. We observed empirically that this improvement enables a significant reduction of the solution time.

3.1.5 Computational results

We start by comparing the five formulations of the deterministic (PCP) in Table 3.2 on instances from the ORLIB [15]. We can see that (F_1) is never able to solve optimally any instance within the time limit of 7200 seconds. For the other formulations, the results vary from one instance to another but in average (F_3) is the slowest and (F_4) the fastest. We will see that the opposite is true for the resolution of the robust problem.

To create instances of $(RPCP)$, we first obtain nominal distances $\{d_{ij}\}_{(i,j)\in V\times U}$ by randomly generating two-dimensional coordinates for the clients and the sites, and nominal clients weights $\{w_i\}_{i\in V}$ randomly taken in $[1000, 2000]$. The box uncertainty set is parameterized by α_1^{ij} and α_2^i randomly generated in either $[0.1, 0.3]$, $[0.4, 0.6]$, or $[0.7, 0.9]$ for each $i \in V$ and each $j \in U$. In a scenario, the distance between client $i \in V$ and site $j \in U$ is in $[d_{ij};\ d_{ij}(1 + \alpha_1)]$, while the weight

Instance			Time (s)				
Name	$\|V\| = \|U\|$	p	F_1	F_2	F_3	F_4	F_5
pmed35	800	5	TL	2116	262	**129**	201
pmed36	800	10	TL	4243	4889	1875	**314**
pmed37	800	80	TL	1681	3922	**1595**	3251
pmed38	900	5	TL	**89**	1235	183	609
pmed39	900	10	TL	1932	**521**	1687	1932
pmed40	900	90	TL	**1419**	6397	1570	2341
		Average	TL	1913	2871	1173	1441

Table 3.2. *Performance comparison of the (PCP) formulations. Symbol TL indicates that the time limit of 7200s is reached. The best result of each instance is represented in bold.*

of client i is in $[w_i(1-\alpha_2);\ w_i(1+\alpha_2)]$. Table 3.3 presents a representative extract of our results on $(RPCP)$ on such instances.

Instance		Time (s)						
		C&CG					B&C	
$\|V\| = \|U\|$	$\alpha_1^{ij},\ \alpha_2^i$	RF_1	RF_2	RF_3	RF_4	RF_5	RF_3	RF_5
15	$\in$[0.1, 0.3]	0.7	0.7	**0.4**	15,5	0.5	**0.4**	**0.4**
15	$\in$[0.4, 0.6]	3.6	5.1	0.8	59.7	1.0	0.7	**0.5**
15	$\in$[0.7, 0.9]	19.6	14.4	1.5	200.3	2.0	**1.0**	**1.0**
25	$\in$[0.1, 0.3]	6.7	4.7	0.9	157.3	1.7	1.1	**0.7**
25	$\in$[0.4, 0.6]	18.6	27.3	6.3	TL	5.5	3.2	**2.0**
25	$\in$[0.7, 0.9]	305.8	94.8	14.6	TL	17.9	**3.9**	4.1
40	$\in$[0.1, 0.3]	213.1	TL	20.6	TL	22.7	9.4	**8.5**
40	$\in$[0.4, 0.6]	TL	TL	389.7	TL	161.7	26.9	**18.1**
40	$\in$[0.7, 0.9]	TL	TL	TL	TL	TL	**71.2**	74.9

Table 3.3. *Results of C&CG and B&C algorithms on randomly generated instances for $(RPCP)$ with $p = 3$. Symbol TL indicates that the time limit of 600s is reached.*

We first note that the difficulty of the problem increases with the bounds on the uncertain parameters $\{\alpha_1^{ij}\}_{(i,j)\in V\times U}$ and $\{\alpha_2^i\}_{i\in V}$. Then, we can see that the C&CG algorithms are significantly slower than the B&C algorithms. Even though formulation (F_3) is one of the least efficient to solve the deterministic (PCP), the branch-and-cut based on (RF_3) is the fastest to solve $(RPCP)$.

In this section we have seen how to model and solve a specific robust problem in order to obtain a solution which cost is minimal in the worst-case scenario. In the next section, we present a more general robust framework in which we are not only interested in the cost of the solution but also on the number of structural modifications that will be required to make the solution feasible if the scenarios occur.

3.2 A new framework for structurally robust solutions

Liebchen et al. [78] introduced a general two-stage framework, called *recoverable robustness*, in which a *nominal solution* x^{nom} and a recovery algorithm A are determined. In this framework, the *nominal objective* (i.e., the objective of the deterministic problem) is optimized and once uncertain scenario ω is revealed, applying A on x^{nom} must enable to obtain a *recovered solution* x^ω (i.e., a solution feasible for scenario ω). The cost incurred when adapting the solution x^{nom} to a scenario is called the *recovery cost*. This cost can correspond to a deterioration of the nominal objective (e.g., the financial cost of adding or delaying trains) or to potential risks related to

a change of solution (e.g., customers missing a train which platform is modified or potential miscommunications to implement the unanticipated changes).

We consider a similar framework and propose for the first time to minimize the recovery cost that we model by a distance between x and x^ω. This is motivated by the fact that in many applications, once the uncertainty is revealed, it can be more important for x^ω to be as structurally similar as possible to x^{nom} than to minimize its nominal objective value. This for example occurs when the nominal solution is implemented on a regular basis – to avoid human errors due to habit – or when the uncertainty is revealed late – since there simply may not be not enough time to make a lot of modifications. Our approach consists in solving a robust problem called the *proactive problem*. We also study the *reactive problem* which consists in adapting a solution to a scenario.

This work has been realized in collaboration with Sourour Elloumi [P2] and is inspired from the PhD thesis of Rémi Lucas [81], [P14] at SNCF that was also co-supervised by Francois Ramond. Our main contributions are:

- the definition of the nominal, the reactive and the proactive problems;
- the characterization of the complexity of the reactive and the proactive problems for two different solution distances and two robust flow problems;
- the identification of a class of proactive problems for which considering a discrete set of uncertain scenarios Ω is equivalent to considering its convex hull $conv(\Omega)$;
- a case study on a railroad planning problem in which we compare the proactive approach to two other approaches from the literature: the anchored approach and the k-distance approach.

3.2.1 Problems definition

We consider an optimization problem in which the set of feasible solutions depends on uncertain parameters that can take any value in an uncertainty set Ω. Consequently, a solution is generally not feasible for all scenarios and may require to be adapted once the uncertainty is revealed. In the deterministic version (P) of this optimization problem, called the *nominal problem*, the uncertain parameters take their nominal value ω^{nom}:

$$(P) \begin{cases} \min & f(x) \\ \text{s.t.} & x \in X(\omega^{nom}) \end{cases}$$

where $X(\omega^{nom})$ is the feasibility set associated with parameters ω^{nom}. The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is called the *nominal objective*.

We model the uncertainty by a discrete set of scenarios that may occur $\Omega = \{\omega_i\}_{i=1}^{|\Omega|}$. To each scenario $\omega_i \in \Omega$ is associated its feasibility set $X(\omega_i)$. Both feasibility sets $X(\omega^{nom})$ and $X(\omega_i)$ can be any subsets of $\mathbb{R}^n$ and they are thus not restricted to linear constraints. Each scenario in Ω corresponds to a possible realization of the uncertainty (e.g., changes in the passenger demands, unavailability of resources, ...).

In this work we consider robust flow problems which can be modeled as MILPs. To represent uncertain flow demands or capacities on the arcs of a graph, we consider constraints with uncertain right-hand side coefficients. In that context, a scenario ω is a vector of $\mathbb{R}^m$ which directly corresponds to the value of these right-hand side coefficients: $X(\omega) = \{x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2} \mid Ax \leqslant \omega\}$.

We now introduce two problems that represent two possible approaches to handle the uncertainty. In the *reactive problem*, the uncertainty has not been anticipated and once scenario ω is revealed, the initially planned solution x^{nom} must be adapted into a reactive solution x^r feasible for scenario ω (i.e., $x^r \in X(\omega)$). In contrast, the *proactive problem* is a robust approach in which a nominal solution and a recovery solution for each scenario in Ω are computed a priori.

Reactive robust solutions

We suppose that the implementation of a solution x^{nom} of the nominal problem (P) is planned and that scenario $\omega \in \Omega$ occurs at operating time (i.e., a few days or hours before its realization). To cause as little disruption as possible, the *reactive problem* provides a *reactive solution* $x^r \in X(\omega)$ which distance to x^{nom} is minimal:

$$P^r(\omega, x^{nom}) \begin{cases} \min & d(x^r, x^{nom}) \\ \text{s.t.} & x^r \in X(\omega) \end{cases}$$

where $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a solution distance that models the recovery cost between x^r and x^{nom}. The reactive problem $P^r(\omega, x^{nom})$ is only considered once the uncertainty is revealed. We now introduce the proactive problem which allows further minimize the recovery cost by anticipating the uncertainties.

Proactive robust solutions

A reactive solution x^r may have a high nominal objective value $f(x^r)$ as f is not taken into account in the reactive problem. Furthermore, the recovery cost may also be high as it has not been anticipated. To address these issues, we introduce the two-stage *proactive problem* whose variables are a *proactive solution* x^p of the nominal problem, set in the first stage, and a recovery solution x^i for each scenario $\omega_i \in \Omega$, set in the second stage.

$$P^p(\Omega, c^*) \begin{cases} \min\limits_{x^p \in X(\omega^{nom})} \max\limits_{\omega_i \in \Omega} \min\limits_{x^i \in X(\omega_i)} d(x^p, x^i) & (3.14) \\ \text{s.t. } f(x^p) = c^* & (3.15) \end{cases}$$

where c^* is the optimal value of the nominal problem (P).

This problem minimizes the maximal recovery cost over all the scenarios while ensuring that x^p is feasible for (P) and that each x^i is feasible for scenario ω_i. Constraint (3.15) ensures that the nominal objective value of x^p is equal to c^*. Consequently, x^p is an optimal solution of the nominal problem which additionally minimizes the maximal recovery cost over Ω.

Both the reactive and the proactive problems require the definition of a solution distance d. We now present the two distances that we considered.

The first one corresponds to the ℓ_1 norm that we call the *distance in values*.

Definition 3.3 *The distance in values between two solutions $x^1 \in \mathbb{R}^n$ and $x^2 \in \mathbb{R}^n$ is*

$$d_{val}(x^1, x^2) = \sum_{j=1}^{n} |x_j^1 - x_j^2|.$$

Depending on the context, d_{val} is not necessarily the most relevant distance. In rail planning, for example, increasing the number of cars in an existing train by one is much less expensive than planning a new train with only one car. This is because creating a train requires checking the availability of a locomotive and agents as well as the compatibility of the new train's schedule with that of other trains. Thus, we also introduce a *distance in structure*. Let $\mathbb{1}_b$ be equal to 1 if and only if condition b is true.

Definition 3.4 *The distance in structure between two solutions $x^1 \in \mathbb{R}^n$ and $x^2 \in \mathbb{R}^n$ is*

$$d_{struct}(x^1, x^2) \sum_{j=1}^{n} |\mathbb{1}_{x_j^1 > 0} - \mathbb{1}_{x_j^2 > 0}|.$$

3.2.2 Complexity results

Our main contribution is the characterization of the complexity of reactive and proactive problems for both distances d_{val} and d_{struct} for two robust flow problems which deterministic versions are polynomial. These two robust problems are detailed in Appendix D.

Theorem 3.2 *For both the min-cost flow problem with uncertain demands and the max-flow problem with uncertain capacities:*

- *the proactive problem with distance d_{val} is strongly $\mathcal{NP}$-hard;*
- *the reactive problem with distance d_{val} is polynomial;*
- *the proactive problem with distance d_{struct} is strongly $\mathcal{NP}$-hard even with 1 scenario;*
- *the reactive problem with distance d_{struct} is strongly $\mathcal{NP}$-hard.*

Sketch of proof - For each of the six problems proved to be $\mathcal{NP}$-hard, we determine a distinct polynomial-time reduction from a known $\mathcal{NP}$-hard problem.

To prove that the complexity of the two reactive problems with distance d_{val} is polynomial, we show that they both correspond to a convex cost flow problem which complexity is known to be polynomial [5]. □

3.2.3 Equivalence on optimization over the discrete set of scenarios set and its convex hull

Let $P(\Omega)$ be a robust problem to which is associated the uncertainty set Ω and let $v(P(\Omega))$ be its optimal value. For many robust optimization problems it has been proved that considering a discrete set of uncertain scenarios Ω is equivalent to considering its convex hull (i.e., that $v(P(\Omega)) = v(P(conv(\Omega)))$ [11, 27, 93]. Such results can make solving $P(conv(\Omega))$ easier by restricting the set of scenarios considered during the resolution.

We first prove that this result holds for a large class of proactive problems $\mathcal{P}_0$.

Proposition 3.1 *Let $\mathcal{P}_0$ be the set of proactive problems in which (i) the distance d is convex, and (ii) the feasibility set of any scenario $\omega \in \mathbb{R}^m$ is a convex set defined as $X(\omega) = \{x \in \mathbb{R}^n \mid h(x) \leqslant \omega\}$, where $h : \mathbb{R}^n \to \mathbb{R}^m$ is a convex function. It holds that $v(P_0(\Omega, c^*)) = v(P_0(conv(\Omega), c^*))$ for all $P_0 \in \mathcal{P}_0$ and all discrete uncertainty sets $\Omega \subset \mathbb{R}^m$.*

Sketch of proof - We first show that the proactive problem can be rewritten as follows:

$$P_0(\Omega, c^*) \begin{cases} \min\limits_{x^p \in X(\omega^{nom})} \max\limits_{\omega_i \in \Omega} g_{x^p}(\omega_i). \\ \text{s.t. } f(x^p) = c^* \end{cases}$$

with $g_{x^p} : \omega \mapsto \min\limits_{x \in X(\omega)} d(x^p, x)$ the function which associates to any scenario $\omega \in conv(\Omega)$ the distance between x^p and its closest solution in $X(\omega)$.

We then prove that g_{x^p} is convex. Finally, since $conv(\Omega)$ is a compact set, the convexity of g_{x_p} ensures that it reaches a maximum on one of its extreme points which, by definition, are all included in Ω. □

Note that $\mathcal{P}_0$ includes in particular proactive problems with distance d_{val} in which the feasibility set $X(\omega)$ of a scenario $\omega \in \Omega$ is defined by linear constraints with uncertain right-hand sides.

We now consider three different classes of proactive problems: (i) $\mathcal{P}_1$ is the same as $\mathcal{P}_0$ but solutions x are additionally imposed to be integers, (ii) $\mathcal{P}_2$ is the same as $\mathcal{P}_0$ except that the distance d is no longer convex, and (iii) $\mathcal{P}_3$ is the same as $\mathcal{P}_0$ but the the feasibility sets are polyhedrons with an uncertain constraint matrix. More precisely, in $\mathcal{P}_3$, the scenarios are matrices in $\mathbb{R}^{n \times m}$ and the feasibility sets are defined by $X : \omega \mapsto \{x \in \mathbb{R}^n \mid \omega x \leqslant b\}$ with $b \in \mathbb{R}^m$. For each of these classes derived from $\mathcal{P}_0$ we show that the results in Proposition 3.1 does not hold anymore.

Proposition 3.2 *In each set $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ there exists an instance P and a discrete uncertainty set Ω, such that $v(P(\Omega, c^*)) < v(P(conv(\Omega), c^*))$.*

Sketch of proof - For each class, we construct an instance P and an uncertainty set reduced to two scenarios: $\Omega = \{\omega_1, \omega_2\}$. We then prove that $v(P(\{\omega_1, \omega_2\}, c^*)) < v(P(\{\frac{1}{2}\omega_1 + \frac{1}{2}\omega_2\}, c^*))$ □

3.2.4 Computational results

In our experiments we design a case study based on the Line Optimization Problem (LOP) which occurs in railway systems with periodic timetables [34,61]. In this problem a line is a path from a departure station to an arrival station with stops in intermediary stations. The objective is to determine which lines to open and to fix their frequencies in order to cover passenger demands while minimizing the costs.

We use this case study to compare the proactive problem with two approaches from the literature: the anchored approach and the k-distance approach. The first approach consists in bounding the nominal objective and maximizing the weights of the anchored variables in the solution [20]. A variable is said to be *anchored* if its value can not be modified once the uncertainty is revealed. This objective can be viewed as a way to limit the recovery cost. The authors considered a scheduling problem with uncertain processing times and show that maximizing the weight of the anchored jobs is polynomial for the box uncertainty set. It is however $\mathcal{NP}$-hard for both the budgeted and the polytope uncertainty sets.

The k-distance approach was introduced and applied to shortest path problems in [33]. In this approach, the recovery actions are limited since at most k new arcs can be used once the uncertainty is revealed. The authors showed that the problem is $\mathcal{NP}$-hard for three distinct uncertainty sets.

For each approach, we consider a discrete uncertainty set Ω and show that the corresponding robust problem can be modeled as a MILP. Similarly to the proactive approach, we fix in the anchored approach the nominal objective of the nominal solution to c^* in order to ease the comparison. The value c^* is obtained by the resolution a priori of the nominal problem (P).

A representative extract of the numerical results obtained for the case study are presented in Table 3.4. Ten scenarios have been generated and three instances have been obtained by considering the first 2, the first 6, and all 10 scenarios. The number of scenarios is reported in the first column. The k-distance approach requires to fix the value of parameter $k \in \mathbb{N}$ which represents the maximal number of differences between the nominal solution and any scenario solution. Since there is no a priori relevant choice for this parameter we set its value to 0, 1, 2, 4 and 10. Consequently, seven lines are associated to each instance, one for the proactive approach, one for the anchored approach, and five for the k-distance approach.

The nominal objective is reported in the third column. Since its optimality is imposed by the proactive and the anchored approach, it is equal to $c^* = 81742$ regardless of the number of scenarios. This objective is larger for the k-distance approach whenever k is small. This highlights a drawback of the k-distance approach which does not allow to constrain the nominal objective of the nominal solution. In other applications, a low value of k could even lead to infeasibilities. Each approach provides a nominal solution as well as a one solution for each considered scenario. From these solutions we deduce the value of the nominal objective, the distance d_{val} and the number of anchored lines. Note that among all solutions with an optimal nominal objective of 81742, the proactive approach returns one with an optimal value d_{val} (e.g., 50 for $|\Omega| = 2$) and the anchored approach returns one with an optimal number of anchored lines (200 for $|\Omega| = 2$). For small values of k, the k-distance is only to improve d_{val} and the number of anchored jobs but at the cost of the nominal objective. For example when no difference is allowed between the nominal and the scenario solutions, all the 210 lines are anchored and the distance d_{val} is null. However, we observe a quick deterioration of these two objectives when k increases showing once again that the choice of parameter k can be challenging. Moreover, the value k is shared by all scenarios while some of them may require less changes than others. This may lead to less suitable scenario solutions than necessary.

The proactive and the anchored approaches do not lead to the same solutions since the recovery cost is not optimal in the anchored approach and the number of anchored lines is not optimal in the proactive approach. These differences are increased with the number of scenarios as it becomes harder to satisfy both objectives simultaneously. Note that the increase in d_{val} provided by the anchored approach is often greater than the decrease in number of variables anchored in the proactive approach. This is due to the fact that anchoring a variable is quite constraining as it requires its value to be identical in the nominal solution and in all the scenario solutions. The advantage is that it guarantees that parts of the nominal solution will not be disrupted. However, this comes at a price in terms of flexibility which is reflected by an increase of the recovery costs d_{val}.

In most cases, resolutions are completed within seconds. However, we notice that the computation time increases with the number of scenarios and even becomes prohibitive for the k-distance approach. For example, for $k = 2$ more than 39 hours were necessary.

$\|\Omega\|$	**Method**	**Nominal objective**	d_{val}	**Number of anchored lines**	**Time (s)**
2	Proactive	**81742**	**50**	192 (-4%)	9
	Anchored	81742	78 (+56%)	**200**	18
	0-distance	89028 (+9%)	0 (-100%)	210 (+5%)	0
	2-distance	84740 (+4%)	19 (-62%)	206 (+3%)	8
	4-distance	82156 (+1%)	43 (-14%)	202 (+1%)	7
	10-distance	81742	96 (+92%)	194 (-3%)	3
6	Proactive	**81742**	**120**	185 (-7%)	22
	Anchored	81742	189 (+58%)	**199**	80
	0-distance	91998 (+13%)	0 (-100%)	210 (+6%)	1
	2-distance	85178 (+4%)	56 (-53%)	201 (+1%)	503
	4-distance	82158 (+1%)	122 (+2%)	195 (-2%)	75
	10-distance	81742	248 (+107%)	170 (-15%)	18
10	Proactive	**81742**	**196**	181 (-9%)	25
	Anchored	81742	327 (+67%)	**198**	500
	0-distance	94308 (+15%)	0 (-100%)	210 (+6%)	0
	2-distance	86428 (+6%)	80 (-59%)	197 (-1%)	141417
	4-distance	83124 (+2%)	153 (-22%)	190 (-4%)	4718
	10-distance	81742	400 (+104%)	158 (-20%)	27

Table 3.4. *Results of all three robust approaches. For a given number of scenarios $|\Omega|$, the percentage in a cell corresponds to the relative change between the cell value and the value in bold in the same column.*

Chapter 4

Learning from the data

The rapid growth of data science has led to the development numerous methods combining Machine Learning (ML) and operational research (OR) that can be regrouped in two categories.

The methods in the first category aims to solve ML problems through OR approaches. This for example includes regression problems [7, 30], dimension reduction [23], interpretable matrix completion [26], and factor analysis [22]. In Section 4.1, we consider supervised classification which is one of the most well-known learning task. We propose new models to improve the solution of the optimal classification tree problem.

The second category of methods uses ML to improve the solution OR problems. Some approaches are for example dedicated to learning good feasible solutions [28,29]. Another consists in learning when a Dantzig-Wolf decomposition should be applied and if several are possible, which one is the best [77]. Several works have also been dedicated to learning good cuts to add in cutting planes and branch-and-cut algorithms [92, 101]. In this context, we introduce in Section 4.2 a reinforcement learning approach to learn how to make branching decision in a B&B.

4.1 New strong formulations to build optimal classification trees

A supervised classification problem considers a dataset $(X_i, y_i)_{i\in\mathcal{I}}$ such that $X_i = (x_{i,j})_{j\in\mathcal{J}} \in \mathbb{R}^{|\mathcal{J}|}$ is the features vector of data $i \in \mathcal{I}$ and $y_i \in \mathcal{K}$ is its associated class. The objective is to determine a classification function $\mathcal{C} : \mathbb{R}^{|\mathcal{J}|} \mapsto \mathcal{K}$, called a *classifier* to best predict the class of new data from their features. To that end, the data are partitioned into two subsets: the *training set* used to train the classifier and the *test set* used for the evaluation. The evaluation consists in computing the percentage of exact predictions over the test set.

The work presented in this article aims to provide classifiers which both have good performances on the training set and are *interpretable*. Interpretability is a concept that is increasingly considered in supervised classification [36] which can be summarized as the ability to explain or to present in understandable terms to a human how a classifier works [45]. There is currently no clear metric to measure interpretability but rather a consensus on which models are (or are not) interpretable [111]. Several reasons explain the rising interest of interpretability. For example, in the General Data Protection Regulation (GDRP), the notion of a right to explanation was introduced [60,107]. Interpretability may also allow users to have more confidence in the results of a classifier which is particularly important when taking sensitive decisions (e.g. medical or legal applications).

Interpretability and prediction performances are usually conflicting goals as the most efficient classifiers tend to be very complex (e.g., deep neural networks). In order to both obtain interpretability and good performances, we consider decision tree classifiers – which are commonly regarded as among the most interpretable – and focus on the exact resolution of the training problem.

A decision tree is an oriented binary tree $\mathcal{T} = (\mathcal{N} \cup \mathcal{L}, A)$ which associates a split function $f_t : \mathbb{R}^{|\mathcal{J}|} \to \{true, false\}$ to each of its internal node $t \in \mathcal{N}$ and a class $k \in \mathcal{K}$ to each of its leaves $\ell \in \mathcal{L}$. A data $i \in \mathcal{I}$ is classified by following a path from the root to a leaf. This path is determined by applying the split functions of each internal node reached to the features vector X_i. Data i follows the left branch of node $t \in \mathcal{N}$ if $f_t(X_i)$ is true and the right branch otherwise. The class predicted by the classifier is the one associated with the leaf reached by data i.

The split functions are generally linear functions $a^\intercal X_i < b$ with $a \in \mathbb{R}^{|\mathcal{J}|}$ and $b \in \mathbb{R}$. When the vectors a are restricted to be binary unit vectors, the tree is said to have *axis-aligned* splits. Otherwise, we say that the splits are *oblique*. Since our contributions are similar for the axis-aligned and the oblique cases, we only focus on the former in this section.

The problem of the construction of an optimal decision tree is $\mathcal{NP}$-complete [69] and the most used approaches are greedy heuristics such as CART [32]. Recently, exact approaches based on MILP have been proposed to solve this problem [24, 105]. However, these formulations quickly become intractable with the size of the dataset. Consequently, to reduce the solution time, other approaches consider less ambitious goals by either restricting the datasets to be binary (i.e., X_i in $\{0,1\}^{|\mathcal{J}|}$ instead of $\mathbb{R}^{|\mathcal{J}|}$) [3, 4, 41, 79] or by developing heuristics approaches [35, 48, 52].

The results presented in this section have been obtained during the PhD of Valentine Huré which I co-supervise with Amélie Lambert [P4]. Our work focuses on improving the resolution of the general case in which the data are not restricted to be binary and the optimality of the solution is guaranteed. Our main contributions are:

- a quadratic formulation based on the model (T) from [24] that we linearize in two different ways;
- an extension of the flow formulation (F_b) from [3] to non-binary datasets;
- the proof that our three linear formulations have a better relaxation than that of (T);
- an iterative algorithm to fit the parameters of our models;
- extensive experimental results in which we show the efficiency of our formulations and of our iterative algorithm.

4.1.1 New formulations

This section presents the improvements we made to formulations (T) and (F_b). For each formulation I describe its sets of variables and the key features that we improved. Due to the limited number of pages allowed for this thesis, the constraints of the formulations are not detailed in this section but in Appendix E.

Without loss of generality, we consider that the value of each feature belongs to $[0, 1]$.

Reinforcement of (T)

The first sets of variables considered in formulation (T) determine the structure of the classifier:

- binary variables $c_{k,\ell}$ is equals to 1 if and only if leaf $\ell \in \mathcal{L}$ predicts class $k \in \mathcal{K}$;
- variables $\{a_{j,t}\}_{(j,t)\in\mathcal{J}\times\mathcal{N}}$ and $\{b_t\}_{t\in\mathcal{N}}$ in $[0, 1]$ are the coefficients of the internal nodes split functions;
- binary variable d_t is equal to 1 if and only if internal node $t \in \mathcal{N}$ performs a split or if leaf $t \in \mathcal{L}$ is reached by a data;

The path of each data $i \in \mathcal{I}$ in the tree is followed through binary variables $\{z_{i,\ell}\}_{(i,\ell)\in\mathcal{I}\times\mathcal{L}}$ equal to 1 if and only if data i reaches leaf ℓ.

Finally, the last sets of variables enable to count the number of misclassifications:

- $N_{k,\ell}$ is equal to the number of data of class $k \in \mathcal{K}$ reaching leaf $\ell \in \mathcal{L}$;
- N_ℓ is equal to the number of data reaching leaf $\ell \in \mathcal{L}$;
- L_ℓ is equal to the number of misclassifications in leaf $\ell \in \mathcal{L}$.

Our reinforcement of (T) is based on a reformulation that removes these last three sets of variables.

Formulation (T) has two conflicting objectives. The first one is the minimization of the misclassifications: $\min \sum_{\ell \in \mathcal{L}} L_\ell$. The second one is the maximization of the interpretability which corresponds to the minimization of the number of internal nodes performing a split: $\min \alpha \sum_{t \in \mathcal{N}} d_t$, with $\alpha \geqslant 0$ a coefficient enabling to adjust the weight given to interpretability. This second objective also enables to avoid *overfitting* which is a common pitfall of supervised classification that occurs when a classifier has very good performances on the training set at the expanse of its performances on the test set.

Formulation (T) has two more parameters: β the minimal number of data reaching a leaf – that also enables to reduce overfitting – and δ the maximal depth of the tree which enables to fix the number of variables and constraints in the model.

We use the following property to define a new formulation of the problem.

Property 4.1 $L_\ell = \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I} \setminus \mathcal{I}_k} c_{k,\ell} z_{i,\ell}$

Proof: $L_\ell = \sum_{k \in \mathcal{K}} c_{k,\ell} \left(N_\ell - N_{k,\ell}\right) = \sum_{k \in \mathcal{K}} (c_{k,\ell} \sum_{i \in \mathcal{I} \setminus \mathcal{I}_k} z_{i,\ell}) = \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I} \setminus \mathcal{I}_k} c_{k,\ell} z_{i,\ell}$ □

Our quadratic formulation (Q) consists in removing from (T) variables $\{L_\ell\}_{\ell \in \mathcal{L}}$ and replacing them in the objective by the quadratic expression of Property 4.1. This also enables to remove variables $\{N_{k\ell}\}_{(k,\ell) \in \mathcal{K} \times \mathcal{L}}$ and $\{N_\ell\}_{\ell \in \mathcal{L}}$ and their corresponding constraints.

We then obtain two linear formulations by linearizing the quadratic objective of (Q) in two different ways:

- Formulation (QF) uses Fortet linearization [54] to replace each product $z_{i,\ell} c_{k,\ell}$ by an auxiliary variable $\theta_{i,k,l}$;
- Formulation (QG) uses Glover's procedure [59] to replace each product $c_{k,\ell} \Big(\sum_{i \in \mathcal{I} \setminus \mathcal{I}_k} z_{i,\ell} \Big)$ by a variable $\Theta_{k,\ell}$.

We remind that $v(P_1)$ denotes the optimal value of a problem (P_1) and that $(\overline{P_2})$ corresponds to the linear relaxation of a MILP (P_2). We now compare the linear relaxation of the three linear formulations (T), (QG) and (QF).

Property 4.2

(i) $v(\overline{QF}) \geqslant v(\overline{QG})$;

(ii) $v(\overline{T}) = 0$;

(iii) $v(\overline{QG}) > v(\overline{T})$ *if* $\alpha > 0$.

Sketch of proof - To prove (i) we consider an optimal solution of $(\overline{QF})$ and show that we can necessarily construct a solution of $(\overline{QG})$ with the same objective value.

We prove that $v(\overline{T}) = 0$ by exhibiting a fractional solution of $(\overline{T})$ in which all variables in $\{L_\ell\}_{\ell \in \mathcal{L}}$ and in $\{d_t\}_{t \in \mathcal{N}}$ are equal to 0.

To prove (iii) we show that a solution of $(\overline{QG})$ with an objective value of 0 would necessarily be infeasible. □

We see in Section 4.1.3 that the numerical results reflect the results in this property: (T) leads to the largest solution time and (QF) to the smallest.

Generalization of (F_b) to non-binary datasets

Formulation (F_b) considers two additional nodes on top of $\mathcal{N} \cup \mathcal{L}$: a source s and a think w. In this formulation, a binary flow is associated to each data from s to w which value is 1 if and only if the data is correctly classified.

On top of variables $\{a_{jt}\}_{(j,t)\in\mathcal{J}\cup\mathcal{N}}$ from (T), it also considers:

- binary variable $u^i_{t_1,t_2}$ equal to 1 if and only if data i is correctly classified and if its flow goes through the arc $(t_1, t_2) \in A$;
- binary variable $g_{k,t}$ equal to 1 if and only if node $t \in \mathcal{N} \cup \mathcal{L}$ predicts class $k \in \mathcal{K}$.

This last set of variables plays the same role as variables $\{c_{k,t}\}_{(k,t)\in\mathcal{K}\cup\mathcal{L}}$ of (T). The only difference is that they are also indexed on $\mathcal{N}$ since in the flow formulation, any node can predict a class.

Since (F_b) is only able to handle binary datasets, the values of variables $\{a_{jt}\}_{j\in\mathcal{J}}$ are sufficient to determine the path of the data at internal node $t \in \mathcal{N}$. Indeed, if $a_{jt} = 1$, a data $i \in \mathcal{I}$ reaching node $t \in \mathcal{N}$, goes to the left if $x_{ij} = 0$ and to the left if $x_{ij} = 1$. To take into account non-binary datasets, we adapt formulation (F_b) into formulation (F) by adding the variables $\{b_t\}_{t\in\mathcal{N}}$ considered in (T) and by adapting the constraints ensuring that the paths of the data in the tree are coherent.

We now prove that (T) and (F) have the same optimal objective values.

Property 4.3 *If $\beta = 0$, $v(T) = v(F)$.*

Sketch of proof - We show that from any optimal solution of (T) we can build a solution of (F) with the same objective value and vice-versa. Note that due to the differences in terms of variables and constraints considered in these two formulations, the construction of the solutions and the satisfaction of the constraints are significantly harder than in Property 4.2 □

We then compare their linear relaxations.

Property 4.4 *If $\alpha > 0$, $\beta > 0$, and $|\mathcal{K}| > 1$, then $v(\overline{F}) > v(\overline{T})$.*

Sketch of proof - Similarly to the proof of Property 4.2, we show that there is no feasible solution of $(\overline{F})$ with an objective value of 0. □

Once again, this is reflected in the numerical results which are better for (F) than for (T). A similar result is also obtained in the case of oblique splits.

4.1.2 Parameters fitting algorithm

Our formulations take the three following parameters as an input: the weight of the second objective α, the minimal number of data in a leaf β and the maximal depth of the tree δ. A fitting algorithm called `TreeTraining` was proposed in [24] to fit the value of these parameters. In this context, the data are split in 3 sets: the training set, the test set and the validation set. The formulation considered is then iteratively solved on the training set in order to obtain one classification tree for different values of the parameters. Afterwards, the trees that are Pareto-dominated are removed. Finally, one of the remaining trees with the best performances on the validation set is selected. Since parameter α is continuous, the algorithm can not iterate on all its possible values. To overcome this, the authors remove the second objective and iterate on the possible values of a parameter $C \in \mathbb{Z}^+$ that bounds the total number of internal nodes that perform a split.

This fitting algorithm can take time as it requires at each step to solve a MILP. To reduce the number of MILPs solved, we observe that the misclassifications of optimal classification trees is a decreasing piecewise constant function of C. Hence, it is not necessary to test each possible value of parameter C to obtain all the non-Pareto-dominated solutions. To reduce the number of iterations, we define a new fitting algorithm named `CompactTreeTraining`. It iterates on the values of C in decreasing order and keep in the model the second objective with a value α

low enough to ensure that an optimal number of misclassifications is obtained. Consequently, among all solutions that minimize the misclassifications, we obtain one that additionally minimize the number of splits. At the next iteration, the value of C is set to $\sum_{t\in\mathcal{N}} d_t^* - 1$ with d_t^* the value of variable d_t for $t \in \mathcal{N}$ in the last computed tree.

We see in the numerical results that this procedure enables to significantly reduce the number of iterations of the algorithm.

4.1.3 Computational results

Formulations comparison

We first compare the CPU times of the direct submission to the `Gurobi` solver of the formulations (T), (Q), (QF), (QG), and (F). We run our experiments on 7 different training sets for each of the following classical datasets: Blood donation, Breast cancer, Car evaluation, Dermatology, Iris, Tic tac toe and Wine whose characteristics are described in Table 4.1. We set parameters to the following values: depths $\delta \in \{2, 3, 4\}$, and $\beta = 0$.

We present in Figure 4.1 the performance profile [44] of the CPU time of the considered formulations. Each curve corresponds to a formulation, where each point of a curve gives, for a given factor τ, the percentage of instances whose CPU time was at most τ times greater than the minimal CPU time within the considered formulations. In particular, for $\tau = 1$, we have the proportion of instances on which the formulation was the best on the criterion.

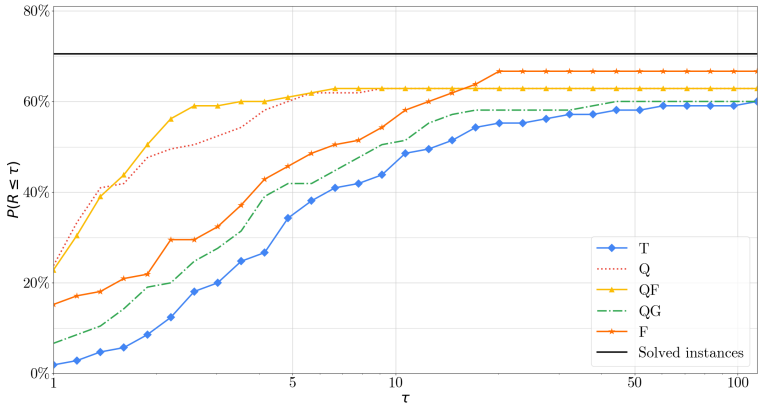

Figure 4.1. *Performance Profile of CPU Times over the 105 considered instances with a time limit of one hour.*

We observe that formulation (T) is the slowest followed by (QG). Formulations (QF) and (Q) are generally faster than (F) which nevertheless solves the most instances to optimality.

In conclusion, the linearization (QG) is not a clear improvement over (T). Formulations (Q), (QF) and (F) significantly outperform (T) in terms of CPU times. Given, the similar performances of (Q) and (QF) we assume that Gurobi uses Fortet's linearization. Consequently, we only focus in the following on formulations (QF) and (F).

Fitting algorithms comparison

We run our experiments on all the datasets whose characteristics are summed up in Table 4.1. For each dataset, we consider the 5 distinct partitions of the data. A partition is described by a training set, a validation set and a test set representing 50%, 25% and 25% of the original datasets, respectively. As in [24], we set the time limit to 1800 seconds.

We start with a comparison of the CPU times and of the number of iterations of algorithm `TreeTraining` run with (T), and of our new algorithm `CompactTreeTraining` run with (QF) and (F). In Figure 4.2 we plot the ratios of CPU time (in blue) and of the number of iterations (in red) for (QF) and (F) compared to (T) for depth $\delta \in \{2, 3, 4\}$. More precisely, each blue

Dataset	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{J}\rvert$	$\lvert\mathcal{K}\rvert$	Dataset	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{J}\rvert$	$\lvert\mathcal{K}\rvert$
Balance scale	625	4	3	Spambase	4601	57	2
Bank marketing 10%	4521	51	2	Statlog satellite	4435	36	7
Car evaluation	1728	6	4	Tic tac toe	958	18	2
Ionosphere	351	34	2	Wine	178	13	3
Iris	150	4	3	Blood transfusion	748	4	2
Monk1	124	6	2	Breast cancer	683	9	2
Monk2	169	6	2	Dermatology	358	34	6
Monk3	122	6	2	Ecoli	336	7	8
Pima Indians diabetes	1151	19	2	German	1000	24	2
Qsar biodegradation	1055	41	2	Haberman	306	3	2
Seismic bumps	2584	18	2	Seeds	210	6	3

Table 4.1. *Datasets considered.*

point corresponds to $\frac{CPU((QF) \text{ or } (F))}{CPU((T))}$ for one dataset. Hence, if a point has a value smaller than 1 it means that our formulation outperforms (T). We observe that `CompactTreeTraining` always reduces the number of iterations in comparison to `TreeTraining`. Moreover, it is often faster, and the reduction of CPU times grows with the depth of the tree. This speed-up is also due to the fact that, at each iteration (QF) and (F) are faster to solve than (T).

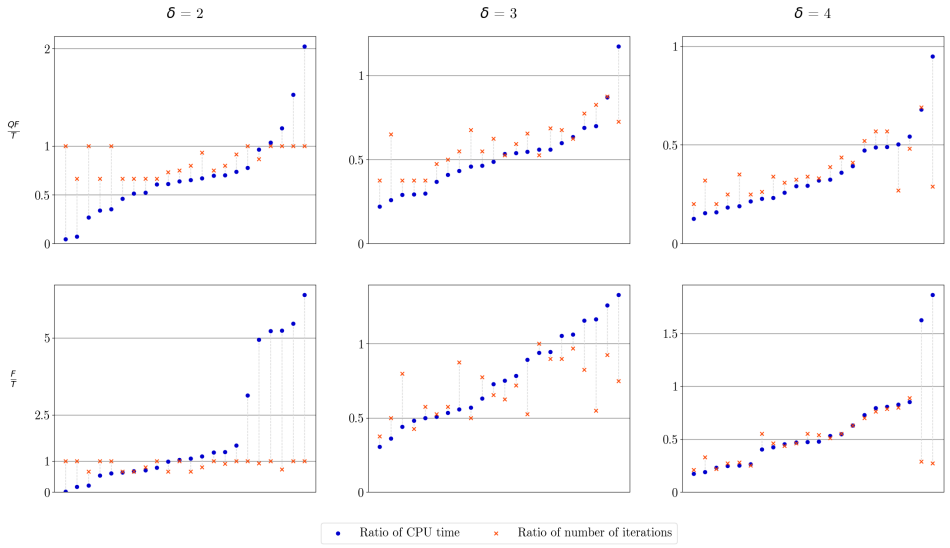

Figure 4.2. *Ratio of CPU times in blue and ratio of number of iterations in red for* `CompactTreeTraining` *with* (QF) *and* (F) *compared to* `TreeTraining` *with* (T).

4.2 A reinforcement learning approach to learn how to branch

At each non-pruned node of a B&B tree considered, a *branching strategy* is used to determine on which integer variable the branching will be performed. Various studies suggest this strategy plays a crucial, if not the most important, role in the building of B&B trees [2,84]. Due to a lack of mathematical understanding of the dynamic nature of such strategies, state-of-the-art solvers use heuristically designed policies, empirically tuned on classic benchmarks from literature (e.g. [76]). The plethora of existing heuristics (see [88] for a survey) reflects the difficulty of the task, and their efficiency heavily depends on the problem to be solved. For this reason, the

use of Machine Learning has been a growing trend over the past few years in the design of the aforementioned policies [21].

When learning strategies in a B&B algorithm, the majority of the literature approaches opt for either imitating [58, 66, 75, 109, 110] or arbitrating between heuristics [12, 42]. In this section, we focus on learning oracle policies, *i.e.* optimal policies with respect to the objective of minimizing the B&B tree size.

The results presented in this section have been obtained during the PhD thesis of Marc Etheve [50], [P13] which was co-supervised by Safia Kedad-Sidhoum as well as two supervisors from EDF: Côme Bissuel and Olivier Juan. The main contributions are:

- the definition of a reinforcement learning framework to learn a B&B branching strategy;
- the definition of a transition function that enables to take into account the structure of the tree;
- the proof that using this transition function and a specific cost model with a dynamic programming approach leads to an oracle strategy;
- the definition of a heuristic Q-learning approach which leads in practice to good results.

4.2.1 Problem definition

Let Π be the set of branching strategies. Our objective is to find a branching strategy called *oracle strategy* that minimizes

$$\min_{\pi \in \Pi} \mathbb{E}_{p \sim \mathcal{L}} \left[|\mathcal{T}^{\pi}(p)| \right] \tag{4.1}$$

where $T^{\pi}(p)$ refers to the tree produced when using strategy π on instance p drawn from a distribution $\mathcal{L}$. In this work $\mathcal{L}$ corresponds to similar MILPs that EDF has to solve on a regular basis. For example, we consider microgrid energy production problems with fixed production units in which only the gas prices and the clients demands vary from one instance to another.

Supervised learning methods do not appear relevant to solve 4.1 as they require labeled data. Indeed, labeling data is not straightforward in this context as it is hard to evaluate the impact of a single branching decision on the size of the B&B tree.

Consequently, we consider Reinforcement Learning (RL) which is an approach dedicated to finding good sequential control strategies that does not require labeled data. Instead policies are learned through trials and errors in order to minimize a cost. This approach emerged in the late 1980s especially with the works of Richard Sutton [99] and Chris Watkins [108] and has been since then a really active and prolific field of research, with many applications in various domains such as games, robotics, or finance.

An RL system is defined by an *agent* that is in a *state* of an *environment* and that must choose an *action* to perform. At time t, applying action a_t in state s_t leads to a new state s_{t+1} through a *transition* at the price of a *cost* $c(s_t, a_t)$.

The choice of the actions taken by the agent is made by a *policy* π_t such that $\pi_t(a|s)$ is the probability of taking action a at state s. RL often assumes the environment to be markovian which gives rise to the notion of Markov Decision Process (MDP) $< \mathcal{S}, \mathcal{A}, T, c, \gamma >$. In the context of our application:

- a state $s \in \mathcal{S}$ corresponds to all the information available when taking a branching decision (e.g., the previous branching decisions, the best feasible solution, ...). Each state s is associated to a B&B node denoted by $\xi(s)$;
- selecting an action $a \in \mathcal{A}$ at a state s corresponds to selecting the variable on which to branch at node $\xi(s)$. Thus, the set of actions $\mathcal{A}$ corresponds to the set of integer variables in the model;
- we consider two transition probabilities T: *trajectory-based* transitions that is the standard transition function used in RL and a new *tree-based* transitions that takes into account the structure of the tree;

- we consider a unitary cost model c (i.e., each node created leads to a cost of 1) and show that it provides better results than considering standard heuristic costs such as strong branching or most fractional;
- the discount factor $\gamma \in [0, 1]$ is used to give a greater weight to the cost of the closest future states when assessing the quality of an action.

Another important strategy in the B&B algorithm is the *node selection strategy* that determines which is the next open node considered after branching. We make the simplifying assumption that this strategy is deterministic (i.e., that taking action $a \in \mathcal{A}$ at state $s \in \mathcal{S}$ always lead to the same subsequent state).

Hypothesis 1 *The node selection strategy of the B&B is deterministic.*

Note that this assumption is not always guaranteed in modern solvers since we do not necessarily have access to all the data they use to select the next visited node. Moreover, their node selection strategy could also rely on non-deterministic heuristics.

We now define the transition function considered in classic RL tasks that associates to a pair (state, action) the next visited state.

Definition 4.1 *The trajectory-based transition function is defined as* $T(s_t, a_t) = s_{t+1}$.

Note that the B&B node $\xi(s_{t+1})$ selected by the node selection strategy can be any open node in the B&B tree. Thus, it does not necessarily correspond to one of the two nodes created when branching on node $\xi(s_t)$.

We focus on branching policies of the form

$$\pi(s) = argmin_{a \in \mathcal{A}} Q^{\pi}(s, a) \tag{4.2}$$

with Q^{π} the discounted sum of costs occurring in state-action pairs visited when applying strategy π from (s, a):

$$Q_{\gamma}^{\pi}(s, a) = \begin{cases} 1 & \text{if } \xi(s) \text{ is a leaf of the B\&B tree} \\ 1 + \gamma Q_{\gamma}^{\pi}(T(s, a), \pi(T(s, a))) & \text{otherwise} \end{cases} \tag{4.3}$$

To sum up, $Q_{\gamma}^{\pi}(s, a)$ enables to evaluate the cost of taking action a at state s and we consider policies that select the action associated to the lowest possible cost.

We define the *value function* associated to Q_{γ}^{π} as $V_{\gamma}^{\pi}(s) = Q_{\gamma}^{\pi}(s, \pi(s))$.

The objective of our RL problem consists in identifying an *optimal policy* π^* with respect to this value function: $V_{\gamma}^{\pi^*} = \min_{\pi \in \Pi} V_{\gamma}^{\pi}(s)$.

4.2.2 Exact solution

It is important to note that an optimal policy is not necessarily an oracle policy since the former minimizes a value function which does not necessarily correspond to minimizing the size of the tree. This is particularly the case if we consider a cost model c based on classical branching heuristics such as most fractional or strong branching [6]. That is the reason why in the following we consider a unit cost model.

We first show that the trajectory-based transition function ensures that an optimal policy is an oracle policy.

Property 4.5 *With trajectory-based transitions and the unit cost model, a policy is optimal for the value function* V_{γ}^{π} *if and only if it is an oracle strategy.*

Sketch of proof - We first show that for $\gamma = 1$, the size of the tree rooted at a given state is equal to its value function.

We then prove that this remains true for any $\gamma \in [0, 1)$ by showing that for any two policies π_1 and π_2, $V_1^{\pi_1}(s) \leqslant V_1^{\pi_2}(s)$ is equivalent to $V_\gamma^{\pi_1}(s) \leqslant V_\gamma^{\pi_2}(s)$. Thus, an optimal policy for $\gamma = 1$ is also optimal for any $\gamma \in [0, 1]$. □

Different exact methods have been introduced to solve RL. The dynamic programming approach called *value iteration* is one of the most well-known (see [100] for a complete introduction).

Theorem 4.6 *Value iteration with trajectory-based transitions and the unit cost model leads to an oracle strategy.*

Sketch of proof - We first show that an optimal value function satisfies a Bellman recursion which can be turned into a dynamic programming operator $\mathcal{B}$.

Then, we prove that $\mathcal{B}$ is a contraction for the L_∞ norm when $\gamma \in [0, 1)$ (i.e., that $||\mathcal{B}V_1 - \mathcal{B}V_2||_\infty \leqslant \gamma||V_1 - V_2||_\infty$ for two arbitrary functions V_1 and V_2).

Finally we prove that there is a unique fixed point for $\mathcal{B}$ that is necessarily an oracle strategy. □

We observe empirically in Section 4.2.4 that despite these results using trajectory-based transitions without taking into account the tree structure is not very efficient. Our explanation for this failure is that this value function is not sufficiently informative and localized. Indeed, after branching at state s, the next visited state may correspond to a node that is not one of the two created by this branching. Consequently, the quality of the action would be evaluated based on the quality of nodes potentially far from $\xi(s)$ in the B&B tree which could make the learning task harder. Consequently, it suffers from a recurrent challenge in RL called *credit assignment* which corresponds to the problem of identifying and thus crediting more the important actions.

To alleviate this problem, we define a second transition function called *tree-based* transitions that takes into account the structure of the tree.

Definition 4.2 *The tree-based transition function is defined as* $T(s_t, a_t) = \{l(s_t, a_t), r(s_t, a_t)\}$ *with* $l(s_t, a_t)$ *(resp.* $r(s_t, a_t)$*), the states that correspond to the left (resp. right) son of the node* $\xi(s_t)$*.*

When considering tree-based transitions, the quality of an action taken at state s is only evaluated based on the states of the nodes in the subtree rooted in $\xi(s)$. Thus, this value function is less dependent on other choices made in the tree, hence more stable and informative which may improve the credit assignment.

This transition function leads to a new definition of the Q-value function:

$$Q_\gamma^\pi(s, a) = c(s, a) + \sum_{s' \in D^\pi(s,a)} \gamma^{d(s')-d(s)} c(s', \pi(s')) \tag{4.4}$$

where $d(s)$ is the depth of $\xi(s)$ in the B&B tree and $D^\pi(s, a)$ is the set states associated to nodes in the subtree rooted in $\xi(s)$.

Unfortunately, we show that, unlike trajectory based transitions, an optimal strategy with tree-based transitions is not necessarily an oracle strategy.

Proposition 4.1 *An optimal policy for the value function associated to tree-based transitions is not necessarily an oracle strategy.*

Sketch of proof - We build a MILP instance in which an optimal solution can only be found in one side of the tree. Then, we design a case where taking a detour (i.e., branching on an unnecessary variable to find the optimum) allows to quickly obtain a strong bound. However, if one only wants to minimize the subtree on the optimal side, this early bound is not found and the global tree is bigger. □

Nevertheless, we prove that the equivalence between oracle and optimal strategies is satisfied if $\gamma = 1$ and if we consider a Depth-First Search (DFS) node selection strategy. The DFS strategy consists in always selecting one of the deepest nodes in the tree.

Proposition 4.2 *With DFS node selection strategy, the unit cost model and $\gamma = 1$, a policy is optimal for the value function associated to tree-based transitions if and only if it is an oracle strategy.*

Sketch of proof - We prove that in DFS, minimizing the tree size is equivalent to minimizing the sub-tree size at each state. □

We also prove that using a dynamic programming approach can lead to an optimal strategy.

Theorem 4.7 *Value iteration with tree-based transitions leads to an optimal strategy for $\gamma \in [0, 0.5)$.*

Sketch of proof - The proof is similar to that of Theorem 4.6. □

Proposition 4.2 is only true for $\gamma = 1$ and Theorem 4.7 is only true for $\gamma \in [0, 0.5)$. Nevertheless, we will see in the numerical experiments that tree-based transitions provide significantly better results than trajectory-based transitions.

Unfortunately, exact methods for RL become quickly intractable with the size of the set of states $\mathcal{S}$. Thus, we cannot apply them in the context of learning a branching strategy as $|\mathcal{S}|$ is exponential in the number of integer variables. Consequently, we opt for a heuristic method.

4.2.3 Approximate solution

When solving a RL problem heuristically, the convergence speed depends on the ability of the learner to search efficiently the state space, which comes by estimating properly the quality of the actions. In this context, the Q-value $Q(s_t, a)$ of action $a \in \mathcal{A}$ at state $s_t \in \mathcal{S}$ is estimated by a surrogate $\hat{Q}_\theta(s_t, a)$ parameterized by a vector θ. The more a state-action pair is visited during the learning, the more its estimation by the surrogate will be accurate. To converge toward an efficient policy, a balance has to be found between exploration of new states and exploitation of states with a good estimated value. This trade-off is called the exploration/exploitation dilemma and exclusively arises in RL by opposition with other forms of learning. This can be handled efficiently by an algorithm called DQN in which the surrogate $\hat{Q}_\theta$ is a neural network. A synthesized version of DQN is shown in Algorithm 10 [87]. It considers $M \in \mathbb{N}$ MILP instances drawn from the considered set of instances $\mathcal{L}$ (Step 4). For each MILP, $T \in \mathbb{N}$ branching decisions are taken with the current policy. To deal with the trade-off between exploration and exploitation, a technique called *ε-greedy exploration* is considered (Step 6). This technique consists in selecting with probability $(1 - \varepsilon)$ the action maximizing $\hat{Q}_\theta$. Otherwise, a random action is selected to enable the exploration of the state-action space. The probability ε may be decreased over time. Considering a surrogate neural network $\hat{Q}_\theta$ can be risky since an update of θ from a specific location in the state space may have an influence on the estimations in any other regions. To overcome this issue, DQN uses experience replay (i.e., updating θ from samples drawn from previously collected tuples of states, actions and costs in Step 9) and batch learning (i.e., considering two parameters θ and θ^- such that θ^- is not updated at each iteration, see

Step 10).

Initialize $\theta = \theta^-$
Create an empty replay buffer B
for each trajectory from 1 to M **do**
    Choose an initial state
    for each element in the trajectory from 1 to T **do**
        Chose an action randomly with probability ε%, otherwise chose $argmax_{a \in A} \hat{Q}_\theta(s_t, a)$
        Add the tuple $(s_t, a, c(s_t, a), s_{t+1})$ to B
        Draw samples from B
        Update θ using the samples drawn from B and θ^- instead of θ in the loss of the neural network $\hat{Q}$
        $\theta^- \leftarrow \theta$ or θ^- periodically

Algorithme 1 : DQN algorithm.

4.2.4 Computational results

In order to obtain a fair comparison of the efficiency of our branching policies, we disable the automatic cuts and the presolve of CPLEX.

The results obtained on energy production problems with the trajectory-based and the tree-based transition functions are presented in Figure 4.3. It represents the average size of the trees found at each step of the training. The tree-based function clearly provides better results. Note that the tree size are still larger than the one produced by CPLEX. However, given the fact that CPLEX branching policy has been improved across decades by numerous researchers, obtaining such close values in a few hours of training is a satisfying result.

Figure 4.3. *Training processes: comparison between tree-based transition (`Uc_tree`) and trajectory-based transitions (`Uc_traj`) for the unitary cost model.*

Fixing the value of the discount factor γ is not a trivial task. We observe empirically that low values of γ tend to make all actions have a similar score that tends towards 1 which makes it harder to learn the most relevant ones. On the other hand, larger values tend to make the value function more volatile since the subtrees can be significantly larger at the root than close to the bottom of a B&B tree. Moreover, the size of the trees can vary a lot due to the fact that all instances are not as difficult to solve and also since the efficiency of the branching strategy improves during training as we can see in Figure 4.3. To solve this dilemma, we introduce the *subtree cost model* defined such that the cost $c(s, a)$ of taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ is equal to the size of the subtree of root $\xi(s)$ and of a given depth $h \in \mathbb{N}$ when action a is taken.

Table 4.2 compares the unit cost model and the subtree cost model on eight sets of energy pro-

Dataset	Unit cost model		Subtree cost model	
	Train	**Test**	**Train**	**Test**
1	+65%	+93%	**+21%**	**+38%**
2	+122%	+157%	**+61%**	**+100%**
3	+222%	+233%	**+6%**	**+22%**
4	+602%	+618%	**+157%**	**+183%**
5	**-33%**	**-35%**	+31%	+36%
6	+2502%	+1585%	**+739%**	**+279%**
7	**+24%**	**+24%**	+335%	+32%
8	+433%	+187%	**+45%**	**+26%**

Table 4.2. *Number of nodes on train and test instances against CPLEX. The performances are displayed in average on train instances over 25 independent training processes. The best results are in bold.*

duction instances with tree-based transitions. Each value corresponds to an average percentage of increase of the tree size compared with CPLEX. Consequently, a negative value indicates that the method provided better average results than CPLEX. The agents are generally less efficient on test instances than on train instances, which was expected, but do not show excessive signs of overfitting. More importantly, we can see that the subtree cost model generally provides significantly more efficient agents.

Chapter 5

Conclusions and perspectives

In this thesis I summarized my main contributions for the exact solution of combinatorial optimization problems modeled as MILPs.

The results presented in Chapter 2 illustrate some typical research work that can be done to make a combinatorial optimization problem more tractable. I address these problems by combining the identification of strong formulations with theoretical results leading to efficient solution algorithms. Thus, for the K-partitioning problem, we extend one of our previous formulations and prove that its linear relaxation is stronger. For the p-median problem, we implement a Benders decomposition and show how to efficiently separate its cuts. In both problems, we identify families of inequalities that are facet-defining and use them to design state-of-the-art solution algorithms. Note that even if these results have been designed for a specific problem, they may nevertheless be applied in broader contexts. For example, in [P5] we show that the facial results obtained for the K-partitioning problem can easily be adapted when seeking at most or at least K clusters.

Since optimal solutions of deterministic problems can be very sensitive to variations of the MILP coefficients, we focus in Chapter 3 on handling uncertainty through robust approaches. We first consider a two-stage robust weighted vertex p-center problem in which the uncertainty lies on the node weights and the distances between clients and sites. We prove that considering a subset of uncertain scenarios is sufficient to obtain an optimal solution and deduce from this result two exact solution algorithms. We also use this result to model the robust problem by adapting five formulations of the deterministic p-center problem from the literature. Secondly, we define a new generic robust framework that minimizes the structural differences of the solutions before and after the uncertainty is revealed. We prove that all the four robust problems considered in this framework are $\mathcal{NP}$-hard. Finally, we highlight the relevance of this new approach through a case study.

To both improve the resolution and expand the scope of mixed integer linear programming, we explore in Chapter 4 links between machine learning and operational research. We first introduce new formulations for the problem of building optimal classification trees which linear relaxations are stronger than the state-of-the-art MILP formulation. We also propose a new algorithm to fit the parameters of these formulations that is twice as fast as the state-of-the-art method. In the second part of this chapter, we propose a reinforcement learning framework to learn how to make branching decisions that minimize the size of the B&B tree. We define a new transition function that enables to take into account the structure of the tree and prove that it leads to an oracle strategy when combined with a dynamic programming algorithm. Since such algorithm is not usable in practice due to the size of the state space, we design a Q-learning heuristic that shows promising results.

I now present my perspectives for each of these three lines of research.

5.1 Perspectives on the solution of hard MILPs

My first research perspective is to improve the clustering of sparse graphs into connected components. Recent works were able to solve such problem to optimality for large graphs with several hundred nodes [67,68]. Our extended formulation (F_{ext}) introduced in Section 2.1 could be improved for this problem. The triangle inequalities is the largest family of constraints in (F_{ext}). To significantly reduce the formulation size, unnecessary triangle inequalities could be removed similarly to [43,85,89]. To scale up, we also plan on designing a matheuristic based on such formulation in which variables would be iteratively fixed depending on their value in an optimal solution of the linear relaxation.

A second perspective would be to take advantage of the recent improvements in the solution of Mixed Integer Quadratic Programs (MIQPs). Several solvers such as CPLEX or Gurobi are now able to solve this type of problems making their resolution even more accessible. I intend to use this opportunity to solve a drone location problem in which the objective is to cover a maximal number of ground units. The modeling of this problem through MILPs reduces the position at which a drone can be positioned to a discrete set which is quite restrictive. However, the use of MIQPs allows us to make the drone locations unconstrained. Unfortunately, the MIQP formulations lead to prohibitive solution times. Consequently, we are currently working on an approach that combines both linear and quadratic formulations to leverage their respective advantages.

Finally, I am currently designing a new exact iterative algorithm for the deterministic p-center problem based on a Benders decomposition, on an initial clustering of the clients, and on an increasingly accurate rounding of the distances between the sites and the clients. In its current version, this algorithm already outperforms the two state of the art methods for this problem.

5.2 Perspectives on dealing with uncertainty

Among all solutions with a minimal cost, the robust approach we introduced in Section 3.2 provides one that additionally minimizes the structural modifications required once the uncertainty is revealed. An interesting perspective would be to study the bi-objective optimization problem which consists in minimizing both the cost and the number of modifications. Instead of one unique solution, this would provide a set of non-dominated solutions which could be very relevant in decision-making contexts.

We proved that all the problems considered within our robust framework are $\mathcal{NP}$-hard even if their deterministic counterparts are polynomial. Thus, it would be very interesting to identify problems that are polynomial. This could be achieved by considering different solution distances or by applying the framework to other applications such as the one considered in Chapter 2.

Finally, since a large number of robust optimization approaches have been developed and studied in the recent years, identifying the most relevant one for a specific use case can be far from trivial. That is why in Section 3.2, we performed a comparison of three robust approaches with similar objectives in the context of a specific railway scheduling problem. An important research perspective would therefore be to generalize this work by a thorough survey of the main existing robust approaches that would include in-depth experiments to assess their respective advantages depending on the type of problem considered.

5.3 Perspectives on learning from the data

The formulation of Machine Learning (ML) problems through MILPs provides a double advantage. First, unlike the vast majority of optimization methods considered in ML, it enables to obtain optimal solutions which could thus provide better performances. Nevertheless, such MILPs must be carefully defined since optimal solutions for a performance objective on a training set is likely to produce overfitting. To avoid this, our MILPs presented in Section 4.1 have a second objective which aims to produce classifiers that are as simple as possible. Moreover,

an iterative algorithm and a validation set are considered to balance these two conflicting objectives. Such mechanisms could be generalized to the modeling of other ML problems. The second advantage provided by the use of MILPs is that they ensure the satisfaction of constraints that can not easily be enforced by most of the ML methods. This is for example the case in sparse regression and sparse reduction of dimension where the number of non-zero components in the solution is bounded [25].

However, the main limitation of the use of MILPs for ML problems is the computation time which can be prohibitive when the size of the datasets increases. That is the reason why defining better formulations and solution algorithms, like we did in Section 4.1 to build optimal classification trees, is a promising line of research. For this same problem we are currently working on a matheuristic based on our new formulations. To reduce the size of the MILPs solved, the algorithm first clusters the data and create a MILP which only considers one data per cluster. The algorithm then iteratively solves this MILP and split clusters until an optimal solution is obtained. This idea of an optimal iterative algorithm based on an initial clustering is also leveraged in the algorithm I am currently developing for the p-center problem (see Section 5.1). The preliminary results provided by these two approaches are very promising and such general technique could be extended to other combinatorial optimization problems. This could open an interesting line of research that both combines algorithmic and theoretical results. Indeed, adapting this approach to other problems first requires to identify stopping conditions that can be proved to guarantee the optimality of the algorithm. Moreover, in order to be computationally efficient, the update of the clusters at each step must be carefully tailored so that they are not too time consuming while also limiting the total number of iterations.

A second perspective I am currently working on that associates ML and mixed integer linear programming is the robustness of neural network. This type of classifiers are currently the most commonly used in supervised classification problems due to their accuracy. Unfortunately, they can be very confident in their predictions even when they are wrong. This over-confidence can be exploited to perform adversarial attacks which consist in imperceptibly perturbing data in order to change their predicted class. This can for example consist in adding a specifically designed sticker on a traffic sign to make it misinterpreted by self-driving cars. We are currently working on modeling the training of robust neural networks by a min-max problem in which the prediction error is minimized for the largest possible perturbation of the data. The design of an efficient method for such problem would have a decisive impact on the reliability and the trust of the users for such classifier. This would be particularly important for sensitive tasks such as those involving human lives.

In continuation of our work presented in Section 4.2, I recently started supervising a new PhD thesis involving the EDF company. We aim to improve the performances of branching strategies obtained through our reinforcement learning algorithm. To achieve this, our initial focus is to identify a more informative state representation so that situations where similar decisions need to be made can more easily be identified. To take advantage of the expertise derived from operational research in the pursuit of optimality in reinforcement learning, we will incorporate domain knowledge into the learning process. Such knowledge could for example correspond to symmetries of the solutions, links between the variables (e.g., variables representing the start-up of a particular machine at consecutive time-steps), or the existence of dominance among the solutions.

Publications

[P1] Z. Ales and S. Elloumi. Compact milp formulations for the p-center problem. In International Symposium on Combinatorial Optimization, pages 14–25. Springer, 2018. https://hal.science/hal-03503279.

[P2] Z. Ales and S. Elloumi. Minimizing recovery cost of network optimization problems. Networks, 81(1):125–151, 2023. https://hal.science/hal-03753311.

[P3] Z. Alès, C. Engelbeen, and R. Figueiredo. Correlation clustering problem under mediation. Accepted in INFORMS Journal on Computing, 2023. https://ensta-paris.hal.science/hal-03503061.

[P4] Z. Alès, V. Huré, and A. Lambert. New optimization models for optimal classification trees. https://hal.science/hal-03865931, November 2022.

[P5] Z. Ales and A. Knippel. An extended edge-representative formulation for the k-partitioning problem. Electronic Notes in Discrete Mathematics, 100(52):333–342, 2016. https://hal.science/hal-04144875.

[P6] Z. Ales and A. Knippel. The k-partitioning problem: Formulations and branch-and-cut. Networks, 76(3):323–349, 2020. https://ensta-paris.hal.science/hal-03428695.

[P7] Z. Ales, A. Knippel, and A. Pauchet. Polyhedral combinatorics of the k-partitioning problem with representative variables. Discrete Applied Mathematics, 211:1–14, 2016. https://hal.science/hal-01759687.

[P8] Z. Ales, T. S. Nguyen, and M. Poss. Minimizing the weighted sum of completion times under processing time uncertainty. Electronic Notes in Discrete Mathematics, 64:15–24, 2018. https://hal.science/hal-01768638.

[P9] Z. Alès, A. Pauchet, and A. Knippel. Extraction and clustering of two-dimensional dialogue patterns. International Journal on Artificial Intelligence Tools, 27(02):1850001, 2018. https://hal.science/hal-02932003.

[P10] Z. Ales, A. Pauchet, A. Knippel, L. Vercouter, and G. Gout. Extraction de motifs dialogiques bidimensionnels. Revue d'Intelligence Artificielle, 29(6):655–683, 2015. https://hal-normandie-univ.archives-ouvertes.fr/hal-02123282.

[P11] C. Duran-Mateluna, Z. Ales, and S. Elloumi. An efficient benders decomposition for the p-median problem. European Journal of Operational Research, 308(1):84–96, 2022. https://hal.science/hal-03450829.

[P12] C. Duran-Mateluna, Z. Ales, S. Elloumi, and N. Jorquera-Bravo. Robust milp formulations for the two-stage weighted vertex p-center problem. Computers & Operations Research, page 106334, 2023. https://hal.science/hal-04146260.

[P13] M. Etheve, Z. Ales, C. Bissuel, O. Juan, and S. Kedad-Sidhoum. Reinforcement learning for variable selection in a branch and bound algorithm. In 17th International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR), Vienna, Austria, September 21–24, pages 176–185. Springer, 2020. https://hal.science/hal-02987320.

[P14] R. Lucas, Z. Ales, S. Elloumi, and F. Ramond. Reducing the adaptation costs of a rolling stock schedule with adaptive solution: The case of demand changes. In 8th International Conference on Railway Operations Modelling and Analysis (ICROMA), Norrköping, Sweden, June 17–20, number 69 in Linköping University Electronic Press, pages 857–876. Linköping University Electronic Press, 2019. https://ensta-paris.hal.science/hal-02428735.

[P15] A. Pauchet, F. Rioult, E. Chanoni, Z. Ales, and O. Serban. Interactive narration requires interaction and emotion. In 5th International Conference on Agents and Artificial Intelligence (ICAART), Barcelona, Spain, February 15–18, pages 527–530, 2013. https://hal.science/hal-01024388.

[P16] R. Regaieg, M. Koubàa, Z. Ales, and T. Aguili. Multi-objective optimization for vm placement in homogeneous and heterogeneous cloud service provider data centers. Computing, 103:1255–1279, 2021. https://ensta-paris.hal.science/hal-03428661.

[P17] B. F. Rosa, M. J. F. Souza, S. r. de Souza, M. F. de Franca Filho, Z. Ales, and P. Michelon. Algorithms for job scheduling problems with distinct time windows and general earliness/tardiness penalties. Computers & Operations Research, 81:203–215, 2017. https://ensta-paris.hal.science/hal-03503054.

[P18] O. Serban, AN. Bersoult, Z. Ales, E. Lebertois, E. Chanoni, F. Rioult, and A. Pauchet. Modélisation de dialogues pour personnage virtuel narrateur. Revue d'Intelligence Artificielle, 28(1):101–130, 2014. https://hal.science/hal-01024530.

Bibliography

[1] T. Achterberg. Scip: solving constraint integer programs. Mathematical Programming Computation, 1:1–41, 2009.

[2] T. Achterberg and R. Wunderling. Mixed integer programming: Analyzing 12 years of progress. Facets of Combinatorial Optimization, pages 449–481, 2013.

[3] S. Aghaei, A. Gomez, and P. Vayanos. Learning Optimal Classification Trees: Strong Max-Flow Formulations, May 2020. https://arxiv.org/abs/2002.09142.

[4] G. Aglin, S. Nijssen, and P. Schaus. Learning Optimal Decision Trees Using Caching Branch-and-Bound Search. AAAI Conference on Artificial Intelligence, 34(04):3146–3153, April 2020. Number: 04.

[5] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. Network flows. Cambridge, Mass.: Alfred P. Sloan School of Management, Massachusetts, 1988.

[6] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Finding cuts in the tsp (a preliminary report). Technical report, Citeseer, 1995. https://dl.acm.org/doi/10.5555/868329.

[7] A. Atamturk and A. Gómez. Safe screening rules for l0-regression from perspective relaxations. In International conference on machine learning, pages 421–430. PMLR, 2020.

[8] A. Atamtürk and M. Zhang. Two-stage robust network flow and design under demand uncertainty. Operations Research, 55(4):662–673, 2007.

[9] I. Averbakh and O. Berman. Minimax regret p-center location on a network with demand uncertainty. Location Science, 5(4):247–254, 1997.

[10] I. Averbakh and O. Oded. Algorithms for the robust 1-center problem on a tree. European Journal of Operational Research, 123(2):292–302, 2000.

[11] J. Ayoub and M. Poss. Decomposition for adjustable robust linear optimization subject to uncertainty polytope. Computational Management Science, 13(2):219–239, 2016.

[12] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik. Learning to branch. In International conference on machine learning, pages 344–353. PMLR, 2018.

[13] O. Baron, J. Milner, and H. Naseraldin. Facility location: A robust optimization approach. Production and Operations Management, 20, 09 2011.

[14] S. Basu, M. Sharma, and P. S. Ghosh. Metaheuristic applications on discrete facility location problems: a survey. OPSEARCH, 52(3):530–561, 2015.

[15] J. E. Beasley. Or-library: Distributing test problems by electronic mail. The Journal of the Operational Research Society, 41(11):1069–1072, 1990.

[16] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski. Robust Optimization. Princeton Series in Applied Mathematics. Princeton University Press, 2009.

[17] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. Mathematical Programming, 99(2):351–376, 2004.

[18] A. Ben-Tal and A. Nemirovski. Robust convex optimization. Mathematics of Operations Research, 23(4):769–805, 1998.

[19] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. Numerische mathematik, 4(1):238–252, 1962.

[20] P. Bendotti, P. Chrétienne, P. Fouilhoux, and A. Pass-Lanneau. The anchor-robust project scheduling problem. Operations Research, 2022.

[21] Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research, 290(2):405–421, 2021.

[22] D. Bertsimas, M. S. Copenhaver, and R. Mazumder. Certifiably optimal low rank factor analysis. Journal of Machine Learning Research, 18(1):907–959, 2017.

[23] D. Bertsimas, R. Cory-Wright, and J. Pauphilet. Solving large-scale sparse pca to certifiable (near) optimality. Journal of Machine Learning Research, 23(13):1–35, 2022.

[24] D. Bertsimas and J. Dunn. Optimal classification trees. Machine Learning, 106(7):1039–1082, July 2017.

[25] D. Bertsimas and J. Dunn. Machine learning under a modern optimization lens. Dynamic Ideas LLC Charlestown, MA, 2019.

[26] D. Bertsimas and M. L. Li. Interpretable matrix completion: A discrete optimization approach. INFORMS Journal on Computing, 2023.

[27] D. Bertsimas and M. Sim. The price of robustness. Operations Research, 52(1):35–53, 2004.

[28] D. Bertsimas and B. Stellato. The voice of optimization. Machine Learning, 110:249–277, 2021.

[29] D. Bertsimas and B. Stellato. Online mixed-integer optimization in milliseconds. INFORMS Journal on Computing, 34(4):2229–2248, 2022.

[30] D. Bertsimas and B. Van Parys. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. The Annals of Statistics, 48(1):300 – 323, 2020.

[31] P. Bonami, V. H. Nguyen, M. Klein, and M. Minoux. On the solution of a graph partitioning problem under capacity constraints. In Combinatorial Optimization: Second International Symposium (ISCO), Athens, Greece, April 19-21, pages 285–296. Springer, 2012.

[32] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. Classification and Regression Trees. Taylor & Francis, January 1984.

[33] C. Büsing. Recoverable robust shortest path problems. Networks, 59(1):181–189, 2012.

[34] M. R. Bussieck, P. Kreuzer, and U. T. Zimmermann. Optimal lines for railway systems. European Journal of Operational Research, 96(1):54–63, 1997.

[35] M. A. Carreira-Perpinan and P. Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. In Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018.

[36] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso. Machine Learning Interpretability: A Survey on Methods and Metrics. Electronics, 8(8):832, July 2019.

[37] G. Cornuejols, G. L. Nemhauser, and L. A. Wolsey. A canonical representation of simple plant location problems and its applications. SIAM Journal on Algebraic Discrete Methods, 1(3):261–272, 1980.

[38] User's manual for cplex. https://www.ibm.com/docs/en/icos/22.1.1?topic=optimizers-users-manual-cplex. Accessed: 2023-06-01.

[39] M. Daskin. Network and discrete location: Models, algorithms and applications. Journal of the Operational Research Society, 48, 01 1996.

[40] M. Demange, V. Gabrel, M. A. Haddad, and C. Murat. A robust p-center problem under pressure to locate shelters in wildfire context. EURO Journal on Computational Optimization, 8:103–139, 2020.

[41] E. Demirović, A. Lukina, E. Hebrard, J. Chan, J. Bailey, C. Leckie, K. Ramamohanarao, and P. J. Stuckey. MurTree: Optimal Decision Trees via Dynamic Programming and Search. Journal of Machine Learning Research, 23(26):1–47, 2022.

[42] G. Di Liberto, S. Kadioglu, K. Leo, and Y. Malitsky. Dash: Dynamic approach for switching heuristics. European Journal of Operational Research, 248(3):943–953, 2016.

[43] T. N. Dinh and M. T. Thai. Toward optimal community detection: From trees to general weighted networks. Internet Mathematics, 11(3):181–200, 2015.

[44] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. Mathematical Programming, 91(2):201–213, January 2002.

[45] F. Doshi-Velez and B. Kim. Towards A Rigorous Science of Interpretable Machine Learning, March 2017.

[46] B. Du and H. Zhou. A robust optimization approach to the multiple allocation p-center facility location problem. Symmetry, 10(11):588, 2018.

[47] B. Du, H. Zhou, and R. Leus. A two-stage robust model for a reliable p-center facility location problem. Applied Mathematical Modelling, 77:99–114, 2020.

[48] J. W. Dunn. Optimal trees for prediction and prescription. Thesis, Massachusetts Institute of Technology, 2018. Accepted: 2018-11-28T15:25:46Z.

[49] S. Elloumi. A tighter formulation of the p-median problem. Journal of Combinatorial Optimization, 19(1):69–83, 2010.

[50] M. Etheve. Solving repeated optimization problems by Machine Learning. PhD thesis, Paris, HESAM, 2021.

[51] N. Fan and P.M. Pardalos. Linear and quadratic programming approaches for the general graph partitioning problem. Journal of Global Optimization, 48(1):57–71, 2010.

[52] M. Firat, G. Crognier, A. F. Gabor, C. A. J. Hurkens, and Y. Zhang. Column generation based heuristic for learning classification trees. Computers & Operations Research, 116:104866, April 2020.

[53] M. Fischetti and J. Jo. Deep neural networks and mixed integer linear optimization. Constraints, 23(3):296–309, 2018.

[54] R. Fortet. L'algebre de Boole et ses applications en recherche operationnelle. Trabajos de Estadistica, 11(2):111–118, June 1960.

[55] E. Gaar and M. Sinnl. A scaleable projection-based branch-and-cut algorithm for the p-center problem. European Journal of Operational Research, 2022.

[56] S. García, M. Labbé, and A. Marín. Solving large p-median problems with a radius formulation. INFORMS Journal on Computing, 23(4):546–556, 2011.

[57] M. R. Garey, R. L. Graham, and D. S. Johnson. Some np-complete geometric problems. In 8th annual ACM symposium on Theory of computing, pages 10–22, 1976.

[58] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi. Exact combinatorial optimization with graph convolutional neural networks. In NeurIPS, 2019.

[59] F. Glover. Improved Linear Integer Programming Formulations of Nonlinear Integer Problems. Management Science, 22:455–460, December 1975.

[60] B. Goodman and S. Flaxman. European Union regulations on algorithmic decision-making and a "right to explanation". AI Magazine, 38(3):50–57, October 2017.

[61] J. W. Goossens, S. Van Hoesel, and L. Kroon. A branch-and-cut approach for solving railway line-planning problems. Transportation Science, 38(3):379–393, 2004.

[62] P. Gupta, M. Gasse, E. Khalil, P. Mudigonda, A. Lodi, and Y. Bengio. Hybrid models for learning to branch. Advances in neural information processing systems, 33:18087–18097, 2020.

[63] Gurobi optimization. https://www.gurobi.com/. Accessed: 2023-06-01.

[64] A. Hasani and H. Mokhtari. Redesign strategies of a comprehensive robust relief network for disaster management. Socio-Economic Planning Sciences, 64:92–102, 2018.

[65] C. Hatice and C. T. Barbaros. Double bound method for solving the p-center location problem. Computers & Operations Research, 40(12):2991–2999, 2013.

[66] H. He, H. Daume III, and J. M. Eisner. Learning to search in branch and bound algorithms. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 3293–3301. Curran Associates, Inc., 2014.

[67] P. Healy, N. Jozefowiez, P. Laroche, F. Marchetti, S. Martin, and Z. Róka. A branch-and-cut algorithm for the connected max-k-cut problem. European Journal of Operational Research, 2023.

[68] V. N. Hung, D. P. Nguyen, and M. Minoux. A model for large scale graph partitioning and efficient upper/lower bound computation via cutting-planes. In 24th National Conference of the Société Francaise de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Rennes, France, February 20–23, 2023.

[69] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. Information Processing Letters, 5(1):15–17, May 1976.

[70] C. Irawan and S. Salhi. Aggregation and non aggregation techniques for large facility location problems: A survey. Yugoslav Journal of Operations Research, 25:1–1, 01 2015.

[71] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. ii: The p-medians. SIAM Journal on Applied Mathematics, 37(3):539–560, 1979.

[72] R Karp. Reducibility among combinatorial problems. Complexity of Computer Computation, pages 85–104, 1972.

[73] L. Kaufman. Partitioning around medoids (program pam). Finding groups in data, 344:68–125, 1990.

[74] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. The Bell system technical journal, 49(2):291–307, 1970.

[75] E. B. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina. Learning to branch in mixed integer programming. In 30th AAAI Conference on Artificial Intelligence, 2016.

[76] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, et al. Miplib 2010. Mathematical Programming Computation, 3(2):103–163, 2011.

[77] M. Kruber, M. E. Lübbecke, and A. Parmentier. Learning when to use a decomposition. In 14th International Conference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR), Padua, Italy, June 5-8, pages 202–210. Springer, 2017.

[78] C. Liebchen, M. Lübbecke, R. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In R. K. Ahuja, R. H. Möhring, and C. D. Zaroliagis, editors, Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems, pages 1–27, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[79] J. Lin, C. Zhong, D. Hu, C. Rudin, and M. Seltzer. Generalized and Scalable Optimal Sparse Decision Trees. In 37th International Conference on Machine Learning, pages 6150–6160. PMLR, November 2020. ISSN: 2640-3498.

[80] C. C. Lu. Robust weighted vertex p-center model considering uncertain data: An application to emergency management. European Journal of Operational Research, 230(1):113–121, 2013.

[81] R. Lucas. Planification adaptative des ressources ferroviaires. PhD thesis, ENSTA Paris, 2020.

[82] J. MacQueen. Classification and analysis of multivariate observations. In 5th Berkeley Symposium on Mathematical Statistics and Probability, pages 281–297. University of California Los Angeles LA USA, 1967.

[83] T. L. Magnanti and R. T. Wong. Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. Operations Research, 29(3):464–484, 1981.

[84] A. Marcos Alvarez, Q. Louveaux, and L. Wehenkel. A supervised machine learning approach to variable branching in branch-and-bound. Technical report, Universite de Liége, 2014. https://orbi.uliege.be/bitstream/2268/167559/1/ecml.pdf.

[85] A. Miyauchi and N. Sukegawa. Redundant constraints in the standard formulation for the clique partitioning problem. Optimization Letters, 9(1):199–207, 2015.

[86] N. Mladenović, J. Brimberg, P. Hansen, and J. A. Moreno-Pérez. The p-median problem: A survey of metaheuristic approaches. European Journal of Operational Research, 2007.

[87] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529, 2015.

[88] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. Discrete Optimization, 19:79–102, 2016.

[89] D. P. Nguyen, M. Minoux, V. H. Nguyen, T. H. Nguyen, and R. Sirdey. Improved compact formulations for a wide class of graph partitioning problems in sparse graphs. Discrete Optimization, 25:175–188, 2017.

[90] J. A. Paul and X. J. Wang. Robust optimization for united states department of agriculture food aid bid allocations. Transportation Research Part E: Logistics and Transportation Review, 82:129–146, 2015.

[91] J. A. Paul and X. J. Wang. Robust location-allocation network design for earthquake preparedness. Transportation research part B: methodological, 119:139–155, 2019.

[92] M. B. Paulus, G. Zarpellon, A. Krause, L. Charlin, and C. Maddison. Learning to cut by looking ahead: Cutting plane selection via imitation learning. In International conference on machine learning, pages 17584–17600. PMLR, 2022.

[93] M. Poss and C. Raack. Affine recourse for the robust network design problem: Between static and dynamic routing. Networks, 61(2):180–198, 2013.

[94] J. Reese. Solution methods for the p-median problem: An annotated bibliography. Networks, 48(3):125–142, 2006.

[95] Gerhard Reinelt. Tsplib - a traveling salesman problem library. ORSA journal on computing, 3(4):376–384, 1991.

[96] M. G. C. Resende and R. F. Werneck. A Hybrid Heuristic for the p-Median Problem. Journal of Heuristics, 10(1):59–88, 2004.

[97] J. B. Sheu. An emergency logistics distribution approach for quick response to urgent relief demand in disasters. Transportation Research Part E: Logistics and Transportation Review, 43(6):687–709, 2007.

[98] A. L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. Operations Research, 21(5):1154–1157, 1973.

[99] R. S. Sutton. Learning to predict by the methods of temporal differences. Machine learning, 3:9–44, 1988.

[100] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

[101] Y. Tang, S. Agrawal, and Y. Faenza. Reinforcement learning for integer programming: Learning to cut. In International conference on machine learning, pages 9367–9376. PMLR, 2020.

[102] A. Trivedi and A. Singh. A hybrid multi-objective decision model for emergency shelter location-relocation projects using fuzzy analytic hierarchy process and goal programming approach. International Journal of Project Management, 35(5):827–840, 2017.

[103] A. Trivedi and A. Singh. Shelter planning for uncertain seismic hazards using multicriteria decision approach: a case of nepal earthquake. Journal of Multi-Criteria Decision Analysis, 26(3-4):99–111, 2019.

[104] T. Tulabandhula and C. Rudin. Robust optimization using machine learning for uncertainty sets, 2014. https://arxiv.org/abs/1407.1097.

[105] S. Verwer and Y. Zhang. Learning Optimal Classification Trees Using a Binary Linear Program Formulation. AAAI Conference on Artificial Intelligence, 33(01):1625–1632, July 2019. Number: 01.

[106] U. Von Luxburg. A tutorial on spectral clustering. Statistics and computing, 17:395–416, 2007.

[107] S. Wachter, B. Mittelstadt, and L. Floridi. Why a Right to Explanation of Automated Decision-Making Does Not Exist in the General Data Protection Regulation. International Data Privacy Law, 7(2):76–99, 2017.

[108] C. J. C. H. Watkins. Learning from delayed rewards. PhD thesis, Cambride University, 1989.

[109] K. Yilmaz and N Yorke-Smith. A study of learning search approximation in mixed integer branch and bound: Node selection in scip. AI, 2(2):150–178, 2021.

[110] Giulia Zarpellon, Jason Jo, Andrea Lodi, and Yoshua Bengio. Parameterizing branch-and-bound search trees to learn branching policies. In AAAI Conference on Artificial Intelligence, volume 35, pages 3931–3939, 2021.

[111] J. Zhou, A. H. Gandomi, F. Chen, and A. Holzinger. Evaluating the Quality of Machine Learning Explanations: A Survey on Methods and Metrics. Electronics, 10(5):593, March 2021.

Appendices

Appendix A

K-clustering formulations

A.1 Edge-representative formulation (F_{er})

In this formulation, binary variable x_{ij} is equal to 1 if and only if nodes i and j are in the same cluster and binary variable r_i is equal to 1 if and only if node i is the node of lowest index in its cluster:

$$
(F_{er}) \begin{cases}
\text{minimize} & \displaystyle\sum_{ij \in E} w_{ij} x_{ij} & & \\
\text{subject to} & x_{ij} + x_{ik} - x_{jk} \leqslant 1 & i \in V\ j,\ k \in V \backslash \{i\},\ j < k & (A.1) \\
& r_j + x_{ij} \leqslant 1 & i, j \in V,\ i < j & (A.2) \\
& \displaystyle r_j + \sum_{i=1}^{j-1} x_{ij} \geqslant 1 & j \in V & (A.3) \\
& \displaystyle\sum_{i=1}^{n} r_i = K & & (A.4) \\
& r_i \in [0,1] & i \in V & \\
& x_{ij} \in \{0,1\} & ij \in E &
\end{cases} .
$$

The triangle inequalities (A.1) ensure that if node i is in the same cluster than nodes j and k, then nodes j and k are also in the same cluster. Constraints (A.2) and (A.3) ensure that the value of r variables are coherent with the value of the x variables. Eventually, Constraint (A.4) enables to obtain exactly K clusters.

A.2 Node-cluster formulations

The two node-cluster formulations that we adapted to the K-clustering problem consider a maximal number of clusters K_{max}, binary variables z_i^k taking value 1 if and only if node $i \in V$ is assigned to cluster $k \in \{1, ..., K_{max}\}$ and binary variables x_{ij} equal to 1 if and only if node i and j are in the same cluster.

A.2.1 Formulation (F_{nc1})

In (F_{nc1}), K_{max} is equal to K:

$$(F_{nc1}) \begin{cases} \text{minimize} & \sum_{ij \in E} w_{ij} x_{ij} & & \\ \text{subject to} & x_{ij} + z_i^k - z_j^k \leqslant 1 & ij \in E,\ k \in \{1, \dots, K_{max}\} & (A.5) \\ & x_{ij} - z_i^k + z_j^k \leqslant 1 & ij \in E,\ k \in \{1, \dots, K_{max}\} & (A.6) \\ & -x_{ij} + z_i^k + z_j^k \leqslant 1 & ij \in E,\ k \in \{1, \dots, K_{max}\} & (A.7) \\ & \sum_{k=1}^{K_{max}} z_i^k = 1 & i \in V & (A.8) \\ & z_i^k = 0 & k > i,\ i \in V,\ k \in \{1, \dots, K_{max}\} & (A.9) \\ & \sum_{i \in V} z_i^k \geqslant 1 & k \in \{1, \dots, K_{max}\} & (A.10) \\ & x_{ij} \in \{0, 1\} & ij \in E & (A.11) \\ & z_i^k \in \{0, 1\} & i \in V,\ k \in \{1, \dots, K_{max}\} & (A.12) \end{cases} .$$

Constraints (A.5), (A.6) and (A.7) are similar to the triangle inequalities and ensure the link between variables x and z. Each node $i \in V$ is in exactly one cluster thanks to Equations (A.8). Equations (A.9) alleviate some of the symmetry by imposing that each node $i \in V$ is not in a cluster whose index is greater than i. Eventually, Constraints (A.10) enforce that all the clusters are non-empty.

A.2.2 Formulation (F_{nc2})

In Formulation (F_{nc2}), K_{max} is equal to $|V|$:

$$(F_{nc2}) \begin{cases} \text{minimize} & \sum_{ij \in E} w_{ij} x_{ij} & & \\ \text{subject to} & (A.5) - (A.9),\ (A.11),\ (A.12) & & \\ & z_j^i \leqslant z_i^i & i, j \in V,\ j > i & (A.13) \\ & \sum_{i \in V} z_i^i = K & & (A.14) \end{cases} .$$

In this variant, Constraints (A.13) ensure that variables z_i^i is equal to 1 if and only if node i is the lowest of its cluster. Consequently, due to Constraint (A.14) exactly K clusters are obtained.

Appendix B

Benders cuts separation algorithm

To separate the Benders cut in $\mathcal{O}(|U|)$, we compute in a pre-processing step the distance matrix $\{\mathcal{S}_{i,r}\}_{(i,r)\in V\times\{1,2,...,|U|\}}$ such that $S_{i,r}$ is the distance between client $i \in V$ and its r^{th} closest site in U. Since index $\tilde{k}_i$ corresponds to the largest index $k \in \{1, 2, ..., K_i\}$ such that $\sum_{j\in U\,:\,d_{ij}\leqslant D_i^k} \overline{y}_j < 1$, the algorithm iteratively removes $\overline{y}_{\mathcal{S}_{ir}}$ from 1 until a value of 0 or less is reached.

Data : a client $i \in V$, distances $\{d_{ij}\}_{(i,j)\in V\times U}$, $p \in \mathbb{N}^*$, ordered distances $\{\mathcal{S}_{i,r}\}_{(i,r)\in V\times\{1,2,...,|U|\}}$, and $\overline{y}$ the current solution of (MP) or $(\overline{MP})$

$\tilde{k}_i \leftarrow 0$
$r \leftarrow 1$
$val \leftarrow 1 - \overline{y}_{\mathcal{S}_{ir}}$
while $val > 0$ *and* $r < M$ **do**
    if $d_{i,\mathcal{S}_{i(r+1)}} > d_{i\mathcal{S}_{ir}}$ **then**
        $\tilde{k}_i \leftarrow \tilde{k}_i + 1$
    $r \leftarrow r + 1$
    $val \leftarrow val - \overline{y}_{\mathcal{S}_{ir}}$
return $\tilde{k}_i$

Algorithme 2 : Computation of index $\tilde{k}_i$ for a given solution $\overline{y}$ of (MP) or $(\overline{MP})$.

Appendix C

Robust weighted p-center problem

C.1 Deterministic formulations of (PCP)

Formulation (F_2) associates one binary variable z^k to each weighted distance in the set $\{w_i d_{ij}\}_{(i,j)\in V\times U}$ [P1]. Let $D^0 < D^1 < \ldots < D^K$ be these sorted weighted distances. Variable z^k is equal to 1 if and only if the radius is greater than or equal to D^k:

$$
(F_2)\begin{cases}
\text{minimize} & D^0 + \sum_{k=1}^{K} \left(D^k - D^{k-1}\right) z^k & & \\
\text{subject to} & z^k + \sum_{j\in U\,:\, w_i d_{ij} < D^k} y_j \geqslant 1 & i \in V\ k \in \{1,2,...,K\} \mid \exists j \in U\ w_i d_{ij} = D^k & \text{(C.1)}\\
& z^k \geqslant z^{k+1} & k \in \{1,2,\ldots,K-1\} & \text{(C.2)}\\
& \sum_{j\in U} y_j = p & & \text{(C.3)}\\
& y_j \in \{0,1\} & j \in U & \\
& z^k \in \{0,1\} & k \in \{1,2,...,K\} &
\end{cases}
$$

Constraints (C.1) indicate that either a client is covered by a center at a distance less than D^k, or that the radius is at least D^k. Constraints (C.2) ensure that if the radius is at least D^{k+1} then it is necessarily at least D^k.

Formulation (F_4) considers a binary variable u^k for all $k \in \{0,1,2,...,K\}$ equal to 1 if and only if the radius is equal to D^k [65]. The solutions of these formulations can be mapped to those of (F_2) by setting $u^0 = 1 - z^1$, $u^k = z^k - z^{k+1}$, and $z^k = 1 - \sum_{k=0}^{K} u^k$.

$$
(F_4)\begin{cases}
\text{minimize} & \sum_{k=0}^{K} D^k u^k & & \\
\text{subject to} & \sum_{j\in U\,:\, w_i d_{ij} \leqslant D^k} y_j \geqslant \sum_{q=0}^{k} u^q & i \in V\ k \in \{0,1,...,K\} & \text{(C.4)}\\
& \sum_{k=0}^{K} u^k = 1 & & \text{(C.5)}\\
& \sum_{j\in U} y_j = p & & \text{(C.6)}\\
& y_j \in \{0,1\} & j \in U & \\
& u^k \in \{0,1\} & k \in \{0,1,...,K\} &
\end{cases}
$$

Constraints (C.4) play a similar role than constraints (C.1). Constraints (C.5) ensure that the radius is only equal to a unique distance.

Finally, Gaar and Sinnl recently presented a formulation obtained from a Benders decomposition of (F_1) [55] which is closely related to a formulation of the *uncapacitated facility location problem* (UFL) from [37] and [83]. This formulation considers a single continuous variable r to represent the radius.

$$
(F_5)\begin{cases}
\text{minimize} & r & & \\
\text{subject to} & r \geqslant w_i d_{ij} - \sum\limits_{j' \in U\,:\, w_i d_{ij'} < w_i d_{ij}} (w_i d_{ij} - w_i d_{ij'}) y_{j'} & i \in V\ j \in U & (C.7)\\
& \sum\limits_{j \in U} y_j = p & & (C.8)\\
& y_j \in \{0,1\} & j \in U &
\end{cases}
$$

Constraints (C.7) ensure that the radius is at least $w_i d_{ij}$ if no site closer to client i is opened. If a unique closer site j' is opened, then r is at least $w_i d_{ij'}$.

C.2 Robust formulations of $(RPCP)$

We adapt (F_2), (F_4) and $(F5)$ to $(RPCP)$ as follows:

$$
(RF_2)\begin{cases}
\text{minimize} & v & & \\
\text{subject to} & v \geqslant D^1_\omega + \sum\limits_{k=2}^{K_\omega} \left(D^k_\omega - D^{k-1}_\omega\right) z^k_\omega - R^*_\omega & \omega \in \overline{\Omega} & (C.9)\\
& z^k_\omega + \sum\limits_{j \in U\,:\, w^\omega_i d^\omega_{ij} < D^k_\omega} y_j \geqslant 1 & \omega \in \overline{\Omega}\ i \in V & \\
& & k \in \{1,2,...,K_\omega\} \mid \exists j \in U\ w^\omega_i d^\omega_{ij} = D^k_\omega & (C.10)\\
& z^k_\omega \geqslant z^{k+1}_\omega & \omega \in \overline{\Omega}\ k \in \{1,2,...,K_\omega - 1\} & (C.11)\\
& \sum\limits_{j \in U} y_j = p & & (C.12)\\
& y_j \in \{0,1\} & j \in U & \\
& z^k_\omega \in \{0,1\} & \omega \in \overline{\Omega}\ k \in \{1,2,...,K_\omega\} &
\end{cases}
$$

$$
(RF_4)\begin{cases}
\text{minimize} & v & & \\
\text{subject to} & v \geqslant \sum\limits_{k=1}^{K_\omega} D^k_\omega u^k_\omega - R^*_\omega & \omega \in \overline{\Omega} & (C.13)\\
& \sum\limits_{j \in U\,:\, w^\omega_i d^\omega_{ij} \leqslant D^k_\omega} y_j \geqslant \sum\limits_{q=1}^{k} u^q_\omega & \omega \in \overline{\Omega}\ i \in V\ k \in \{1,2,...,K_\omega\} & (C.14)\\
& \sum\limits_{k=1}^{K_\omega} u^k_\omega = 1 & \omega \in \overline{\Omega} & (C.15)\\
& \sum\limits_{j \in U} y_j = p & & (C.16)\\
& y_j \in \{0,1\} & j \in U & \\
& u^k_\omega \in \{0,1\} & \omega \in \overline{\Omega}\ k \in \{1,2,...,K_\omega\} &
\end{cases}
$$

$$
(RF5)\begin{cases}
\text{minimize} & v & & & \\
\text{subject to} & v \geqslant w_i^\omega d_{ij}^\omega - \sum_{j' \in U\,:\, w_i^\omega d_{ij'}^\omega < w_i^\omega d_{ij}^\omega} (w_i^\omega d_{ij}^\omega - w_i^\omega d_{ij'}^\omega) y_{j'} - R_\omega^* & \omega \in \overline{\Omega}\ i \in V\ j \in U & (C.17) \\
& \sum_{j \in U} y_j = p & & (C.18) \\
& y_j \in \{0,1\} & j \in U &
\end{cases}
$$

The first difference between these robust formulations and their deterministic counterparts is that to each scenario $\omega \in \overline{\Omega}$ is associated a set of variables $\{z_\omega^k\}_{k \in \{1,2,\ldots,K_\omega\}}$ in (RF_2), and a set of variables $\{u_\omega^k\}_{k \in \{1,2,\ldots,K_\omega\}}$ in (RF_4). Moreover, to minimize the regret instead of the radius, the term R_ω^* is subtracted from the right-hand side of Constraints (C.9), (C.13), and (C.17).

C.3 Column and constraint generation algorithm

Our method is detailed in Algorithm 3. Formulation (RF) can be any of our five robust formulations (RF_1), (RF_2), (RF_3), (RF_4), or (RF_5).

At each iteration, a solution (y, v) which satisfies all the scenarios currently in $\overline{\Omega}$ is obtained by solving (RF) (Step 3). If the solution does not satisfy one of the scenarios $\{\omega_i(y)\}_{i \in V}$ (Step 9), the most violated scenario is added to $\overline{\Omega}$ (Step 13). When no violated scenario is found, an optimal solution is returned.

The value of the optimal radius considering a scenario $w_i(x)$ can be calculated by solving a deterministic (PCP) (Step 7). Note that the radius associated with a feasible solution x in a scenario w_i can be obtained quickly as it only requires to determine the distance between each client and its closest opened site in J_y (Step 8).

input : An instance and a robust formulation (RF) of $(RPCP)$
output : An optimal solution y and its regret v.

1 $v \leftarrow 0$, $\overline{\Omega} \leftarrow \varnothing$, $isOptimal \leftarrow false$
2 **while** isOptimal = false **do**
3 $(y, v) \leftarrow$ solve (RF) with scenarios $\overline{\Omega}$
4 $isOptimal \leftarrow true$
5 $\overline{\omega} \leftarrow \varnothing$
6 **for** $i \in V$ **do**
7 $R^* \leftarrow$ optimal radius of the deterministic (PCP) for scenario $\omega_i(y)$
8 $R \leftarrow$ radius of y in scenario $\omega_i(y)$
9 **if** $R - R^* > v$ **then**
10 $isOptimal \leftarrow false$
11 $v \leftarrow R - R^*$
12 $\overline{\omega} \leftarrow \omega_i(y)$
13 $\overline{\Omega} \leftarrow \overline{\Omega} \cup \{\overline{\omega}\}$
14 **return** y and v

Algorithme 3 : *Column-and-constraint generation* algorithm

Appendix D

Minimizing the recovery cost

D.1 Min-cost flow problems definition

The deterministic min-cost flow problem can be stated as:

MIN-COST FLOW PROBLEM
Input: A digraph $G = (V, A)$ with for each arc $a \in A$ a demand $\omega_a^{nom} \in \mathbb{N}$, a capacity $u_a \in \mathbb{N}$ and a unitary cost $c_a \in \mathbb{R}^+$ and for each vertex $v \in V$ a node demands $b_v \in \mathbb{Z}$ ($b_v > 0$ if v is a *supply node*, $b_v < 0$ if v is a *demand node* and $b_v = 0$ if v is a *transshipment node*).
Output: Find an integer flow with minimal cost.

For a node $v \in V$, let $\delta^-(v)$ and $\delta^+(v)$ be the set of predecessors and successors of v in G, respectively. An integer flow $f \in \mathbb{N}^{|A|}$ is feasible for this problem if it satisfies all the arc demands and capacities (i.e., $f_a \in [\omega_a^{nom}, u_a]\ \forall a \in A$) and the node demands (i.e., $\sum_{u \in \delta^-(v)} f_{uv} - \sum_{u \in \delta^+(v)} f_{vu} = b_v\ \forall v \in V$).

We assume that the uncertainties are on the arc demands ω^{nom}. Consequently, each scenario $\omega \in \Omega$ is a vector of arc demands in $\mathbb{N}^{|A|}$ and $X(\omega)$ represents the set of feasible integer flows when the arc demands are set to ω. The corresponding reactive and proactive problems can be stated as follows:

REACTIVE MIN-COST FLOW PROBLEM
Input: A min-cost flow problem, one of its solutions $f^{nom} \in X(\omega^{nom})$, a distance $d : \mathbb{N}^{|A|} \times \mathbb{N}^{|A|} \to \mathbb{R}^+$ and arc demands $\omega \in \mathbb{N}^{|A|}$.
Output: Find a reactive flow $f^r \in X(\omega)$ which minimizes $d(f^{nom}, f^r)$.

PROACTIVE MIN-COST FLOW PROBLEM
Input: A min-cost flow problem with demands $\omega^{nom} \in \mathbb{N}^{|A|}$, its optimal flow value c^*, a distance $d : \mathbb{N}^{|A|} \times \mathbb{N}^{|A|} \to \mathbb{R}^+$, and a discrete set of scenarios $\Omega \subset \mathbb{N}^{|A|}$.
Output: Find a proactive flow $f^p \in X(\omega^{nom})$ and a flow $f^\omega \in X(\omega)$ for each scenario $\omega \in \Omega$ which minimize $\sum_{\omega \in \Omega} d(f^p, f^\omega)$.

D.2 Max-flow problems definitions

The discrete max-flow problem can be stated as:

MAX-FLOW PROBLEM

Input: A digraph $G = (V, A)$ with a source $s \in V$, a sink $t \in V$ and capacities $\omega_a^{nom} \in \mathbb{N}$ on the flow of each arc $a \in A$.

Output: Find an integer flow with maximum value.

We consider max-flow problems with uncertainties on the capacities. Consequently, each scenario $\omega \in \Omega$ is a vector of arc capacities in $\mathbb{N}^{|A|}$. Note that the associated reactive and proactive problems are not particular cases of the ones considered in the previous section as the uncertainty is not on the arc demands and $X(\omega)$ represents the set of feasible integer flows when the arc capacities are set to ω. The corresponding reactive and proactive problems can be stated as follows:

REACTIVE MAX-FLOW PROBLEM

Input: A max-flow problem, one of its solutions $f^{nom} \in X(\omega^{nom})$, a distance $d : \mathbb{N}^{|A|} \times \mathbb{N}^{|A|} \to \mathbb{R}^+$ and arc capacities $\omega \in \mathbb{N}^{|A|}$.

Output: Find a reactive flow $f^r \in X(\omega)$ which minimizes $d(f^{nom}, f^r)$.

PROACTIVE MAX-FLOW PROBLEM

Input: A max-flow problem with capacities $\omega^{nom} \in \mathbb{N}^{|A|}$, its optimal flow value c^*, a distance $d : \mathbb{N}^{|A|} \times \mathbb{N}^{|A|} \to \mathbb{R}^+$ and a discrete set of scenarios $U \subset \mathbb{N}^{|A|}$.

Output: Find a proactive flow $f^p \in X(\omega^{nom})$ and a flow $f^\omega \in X(\omega)$ for each scenario $\omega \in \Omega$ which minimize $\sum_{\omega \in \Omega} d(f^p, f^\omega)$.

Appendix E

Building optimal classification trees

E.1 Formulations

Since the variables used in the formulations are detailed in Section 4.1.1, we do not recall them in this section.

In the following, we denote by $r \in \mathcal{N}$ the root of the tree, and by $a(t)$ the direct ancestor of node $t \in \mathcal{N} \backslash \{r\}$. Let P_ℓ be the path from r to a leaf $\ell \in \mathcal{L}$, we also denote by $A_L(\ell)$ and $A_R(\ell)$, the subsets of P_ℓ whose left and right child is in P_ℓ, respectively.

E.1.1 Formulation (F)

The formulation (F) from [3] considers a digraph $\mathcal{G} = (\mathcal{N} \cup \mathcal{L} \cup \{s, w\}, \mathcal{E} = A \cup \{(s, r), (t, w)_{t \in \mathcal{N} \cup \mathcal{L}}\})$ with A the arcs of the binary tree:

$$
(F^b) \begin{cases}
\max \sum_{i \in \mathcal{I}} u^i_{s,r} - \alpha \sum_{t \in \mathcal{N}} \sum_{j \in \mathcal{J}} a_{j,t} & \\
\text{s.t.} \sum_{j \in \mathcal{J}} a_{j,t} + \sum_{k \in \mathcal{K}} g_{k,t} = 1 & t \in \mathcal{N} \quad \text{(E.1)} \\
\sum_{k \in \mathcal{K}} g_{k,\ell} = 1 & \ell \in \mathcal{L} \quad \text{(E.2)} \\
u^i_{a(t),t} = u^i_{t,l(t)} + u^i_{t,r(t)} + u^i_{t,w} & t \in \mathcal{N}\ i \in \mathcal{I} \quad \text{(E.3)} \\
u^i_{a(\ell),\ell} = u^i_{\ell,w} & \ell \in \mathcal{L}\ i \in \mathcal{I} \quad \text{(E.4)} \\
u^i_{t,l(t)} \leqslant \sum_{j \in \mathcal{J}: x_{i,j}=0} a_{j,t} & t \in \mathcal{N}\ i \in \mathcal{I} \quad \text{(E.5)} \\
u^i_{t,r(t)} \leqslant \sum_{j \in \mathcal{J}: x_{i,j}=1} a_{j,t} & t \in \mathcal{N}\ i \in \mathcal{I} \quad \text{(E.6)} \\
u^i_{t,w} \leqslant g_{y_i,t} & i \in \mathcal{I}\ t \in \mathcal{N} \cup \mathcal{L} \quad \text{(E.7)} \\
a_{j,t} \in \{0, 1\} & t \in \mathcal{N}\ j \in \mathcal{J} \\
g_{k,t} \in \{0, 1\} & t \in \mathcal{N} \cup \mathcal{L}\ k \in \mathcal{K} \\
u^i_{t,t'} \in \{0, 1\} & i \in \mathcal{I}\ (t, t') \in \mathcal{E}
\end{cases}
$$

Constraints (E.1) ensure that each internal node either performs a split or predicts a class and Constraints (E.2) that exactly one class is assigned to each leaf. Constraints (E.3) and (E.4) impose the flow conservation. Constraints (E.5) and (E.6) ensure the consistency of the split functions. Finally, Constraints (E.7) impose that the flow of a misclassified data is null.

E.1.2 Formulation (T)

A difficulty in the modelisation of the problem on non-binary datasets is that if a data i assigned to leaf ℓ the strict inequalities $\sum_{j\in\mathcal{J}} a_{j,t_l} x_{i,j} < b_{t_l}$ for all $t_l \in A_L(\ell)$ must be true. To satisfy these strict inequalities in a MILP, the authors of formulation (T) [24] consider for each feature $j \in \mathcal{J}$ the smallest positive interval between values of the training data on feature j: $\mu_j = \min_{i_1,i_2\in\mathcal{I}^2}\{|x_{i_1,j} - x_{i_2,j}|, \text{ s.t. } x_{i_1,j} \neq x_{i_2,j}\}$. Consequently, the constraint $x_{i,j} < b_t$ can be replaced by $x_{i,j} + \mu_j \leqslant b_t$. To strengthen the constraints, they also consider $\mu^- = \min_{j\in\mathcal{J}}(\mu_j)$ and $\mu^+ = \max_{j\in\mathcal{J}}(\mu_j)$.

$$
(T)\begin{cases}
\min \displaystyle\sum_{\ell\in\mathcal{L}} L_\ell + \alpha \sum_{t\in\mathcal{N}} d_t & \\
\text{s.t.} \displaystyle\sum_{j\in\mathcal{J}} a_{j,t} = d_t & t\in\mathcal{N} \quad \text{(E.8)}\\
0 \leqslant b_t \leqslant d_t & t\in\mathcal{N} \quad \text{(E.9)}\\
d_t \leqslant d_{a(t)} & t\in\mathcal{N}\backslash\{r\} \quad \text{(E.10)}\\
\displaystyle\sum_{k\in\mathcal{K}} c_{k,\ell} = d_\ell & \ell\in\mathcal{L} \quad \text{(E.11)}\\
\displaystyle\sum_{i\in\mathcal{I}} z_{i,\ell} \geqslant \beta d_\ell & \ell\in\mathcal{L} \quad \text{(E.12)}\\
z_{i,\ell} \leqslant d_\ell & \ell\in\mathcal{L}\ i\in\mathcal{I} \quad \text{(E.13)}\\
\displaystyle\sum_{\ell\in\mathcal{L}} z_{i,\ell} = 1 & i\in\mathcal{I} \quad \text{(E.14)}\\
\displaystyle\sum_{j\in\mathcal{J}} a_{j,t}\left(x_{i,j} + \mu_j - \mu^-\right) + \mu^- \leqslant b_t + (1+\mu^+)(1-z_{i,\ell}) & i\in\mathcal{I}\ \ell\in\mathcal{L}\ t\in A_L(\ell) \quad \text{(E.15)}\\
\displaystyle\sum_{j\in\mathcal{J}} a_{j,t} x_{i,j} \geqslant b_t - (1-z_{i,\ell}) & i\in\mathcal{I}\ \ell\in\mathcal{L}\ t\in A_R(\ell) \quad \text{(E.16)}\\
N_{k,\ell} = \displaystyle\sum_{i\in\mathcal{I}\,|\,y_i=k} z_{i,\ell} & \ell\in\mathcal{L}\ k\in\mathcal{K} \quad \text{(E.17)}\\
N_\ell = \displaystyle\sum_{i\in\mathcal{I}} z_{i,\ell} & \ell\in\mathcal{L} \quad \text{(E.18)}\\
L_\ell \geqslant N_\ell - N_{k,\ell} - |\mathcal{I}|(1-c_{k,\ell}) & \ell\in\mathcal{L}\ k\in\mathcal{K} \quad \text{(E.19)}\\
L_\ell \leqslant N_\ell - N_{k,\ell} + |\mathcal{I}| c_{k,\ell} & \ell\in\mathcal{L}\ k\in\mathcal{K} \quad \text{(E.20)}\\
L_\ell \geqslant 0 & \ell\in\mathcal{L} \quad \text{(E.21)}\\
a_{j,t}\in\{0,1\} & t\in\mathcal{N}\ j\in\mathcal{J} \quad \text{(E.22)}\\
c_{k,\ell}\in\{0,1\},\ d_\ell\in\{0,1\},\ z_{i,\ell}\in\{0,1\} & \ell\in\mathcal{L}\ k\in\mathcal{K}\ i\in\mathcal{I} \quad \text{(E.23)}\\
d_t\in\{0,1\} & t\in\mathcal{N}\cup\mathcal{L} \quad \text{(E.24)}
\end{cases}
$$

Constraints (E.8) and (E.9) ensure that the variables defining the split function of $t \in \mathcal{N}$ are null if t does not perform a split. Constraints (E.10) prevent node t from performing a split if its ancestor does not perform one. Constraints (E.11) ensure that a class is assigned to leaf ℓ if and only if it is reached by a data. The second part of the model follows the path of the data in the tree. Constraints (E.12) and (E.13) ensure that the number of data reaching a leaf is not in $\{1, ..., \beta - 1\}$. Constraints (E.14) impose that each data is assigned to exactly one leaf, and Constraints (E.15) - (E.16) that the path of each data is consistent with the split functions. The last set of constraints counts the number of misclassifications. Constraints (E.17) and (E.18) fix the values of variables $\{N_{k,\ell}\}_{(k,\ell)\in\mathcal{K}\cup\mathcal{L}}$ and $\{N_\ell\}_{\ell\in\mathcal{L}}$ which then enables to fix the value of $\{L_\ell\}_{\ell\in\mathcal{L}}$ through Constraints (E.19) to (E.21).

E.1.3 Our new formulations

We first present our quadratic formulation (Q) and its two linearizations:

$$(Q)\begin{cases}\min & \sum_{\ell\in\mathcal{L}}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} c_{k,\ell}z_{i,\ell} + \alpha\sum_{t\in\mathcal{N}} d_t & \\ \text{s.t.} & (E.8)-(E.16),(E.22)-(E.24) & \\ & l_\ell \leqslant d_t & \ell\in\mathcal{L}\ t\in A_L(\ell)\end{cases} \tag{E.25}$$

Note that Constraints (E.25) are not required but strengthen the formulation.

$$(QF)\begin{cases}\min & \sum_{\ell\in\mathcal{L}}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} \theta_{i,k,\ell} + \alpha\sum_{\ell\in\mathcal{N}} d_\ell & \\ \text{s.t.} & (E.8)-(E.16),(E.22)-(E.24),(E.25) & \\ & \theta_{i,k,\ell} \geqslant c_{k,\ell} + z_{i,\ell} - 1 & i\in\mathcal{I}\setminus\mathcal{I}_k,\ k\in\mathcal{K}\ \ell\in\mathcal{L} \quad (E.26)\\ & \theta_{i,k,\ell} \geqslant 0 & i\in\mathcal{I}\setminus\mathcal{I}_k,\ k\in\mathcal{K}\ \ell\in\mathcal{L} \quad (E.27)\end{cases}$$

Constraints (E.26) and (E.27) correspond to Fortet's linearization. They ensure that $\theta_{i,k,l}$ is necessarily equal to $c_{k,\ell}z_{i,\ell}$.

$$(QG)\begin{cases}\min & \sum_{k\in K}\sum_{\ell\in\mathcal{L}}\Theta_{k,\ell} + \alpha\sum_{t\in\mathcal{N}} d_t & \\ \text{s.t.} & (E.8)-(E.16),(E.22)-(E.24),(E.25) & \\ & \Theta_{k,\ell} \geqslant 0 & k\in\mathcal{K}\ \ell\in\mathcal{L} \quad (E.28)\\ & \Theta_{k,\ell} \geqslant \sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell} - |\mathcal{I}\setminus\mathcal{I}_k|(1-c_{k,\ell}) & k\in\mathcal{K}\ \ell\in\mathcal{L} \quad (E.29)\end{cases}$$

Constraints (E.28) and (E.29) correspond to Glover's procedure. They ensure that $\Theta_{k,l}$ is equal to $c_{k,\ell}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell}$.

Finally, we present our generalisation (F) of the flow formulation (F^b) to non-binary datasets:

$$(F)\begin{cases}\min & |\mathcal{I}| - \sum_{i\in\mathcal{I}} u^i_{s,r} + \alpha\sum_{t\in\mathcal{N}}\sum_{j\in\mathcal{J}} a_{j,t} & \\ \text{s.t.} & (E.1)-(E.4),(E.7),(E.22)-(E.24) & \\ & 0 \leqslant b_t \leqslant \sum_{j\in\mathcal{J}} a_{j,t} & t\in\mathcal{N} \quad (E.30)\\ & \sum_{j\in\mathcal{J}} a_{j,t}\left(x_{i,j}+\mu_j-\mu^-\right)+\mu^- \leqslant b_t + (1+\mu^+)(1-u^i_{t\,l(t)}) & t\in\mathcal{N}\ i\in\mathcal{I} \quad (E.31)\\ & \sum_{j\in\mathcal{J}} a_{j,t}x_{i,j} \geqslant b_t - (1-u^i_{t,r(t)}) & t\in\mathcal{N}\ i\in\mathcal{I} \quad (E.32)\\ & u^i_{t,l(t)} \leqslant \sum_{j\in\mathcal{J}} a_{j,t} & i\in\mathcal{I}\ t\in\mathcal{N} \quad (E.33)\\ & u^i_{t,r(t)} \leqslant \sum_{j\in\mathcal{J}} a_{j,t} & i\in\mathcal{I}\ t\in\mathcal{N} \quad (E.34)\\ & a_{j,t}\in\{0,1\} & t\in\mathcal{N}\ j\in\mathcal{J} \quad (E.35)\end{cases}$$

To ease the comparison with other formulations, we replace the objective by a minimization rather than a maximization. Constraints (E.31) and (E.32) model the split functions. They are obtained from Constraints (E.15) and (E.16) of (T) by replacing variable $z_{i,\ell}$ by either variable $u^i_{t,\ell(t)}$ or $u^i_{t,r(t)}$. Finally, we adapt the capacity constraints (E.5) and (E.6) into Constraints (E.33) and (E.34).