Optimisation discrète robuste en présence d'incertitude ellipsoïdale

Chifaa Dahik

Thèse de doctorat de l'établissement
Université Bourgogne Franche-Comté
preparée à l'Université de Franche-Comté

École doctorale n°37
Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Informatique

Par

Chifaa DAHIK

Optimisation Discrète Robuste En Présence D'incertitude Ellipsoïdale

Thèse présentée et soutenue à Besançon, le 29 Novembre 2021

Composition du Jury :

Zeina AL MASRY	Maître de conférence à l'ENSMM à Besançon	Encadrante de thèse
Stéphane CHRETIEN	Professeur à l'Université de Lyon 2	Examinateur
Clément DOMBRY	Professeur à l'Université de Franche-Comté	Président
Imed KACEM	Professeur à l'Université de Lorraine	Rapporteur
Kim Thang NGUYEN	Maître de conférence à l'Université de Paris Saclay	Examinateur
Jean-marc NICOD	Professeur à l'ENSMM à Besançon	Directeur de thèse
Landy RABEHASAINA	Maître de conférence à l'Université de Franche-Comté	Co-directeur de thèse
Andréa Cynthia SANTOS	Professeur à l'Université Le Havre	Rapportrice

Ph.D. thesis of the University Bourgogne Franche-Comté prepared at the University of Franche-Comté

Doctoral school n° 37

Engineering Sciences and Microtechnologies

Ph.D in Computer Science

by

Chifaa DAHIK

Robust Discrete Optimization Under Ellipsoidal Uncertainty

Thesis defended publicly on November 29, 2021, in Besançon

Composition of jury :

Zeina AL MASRY	Associate professor at ENSMM Besançon	Supervisor
Stéphane CHRETIEN	Professor at Université de Lyon 2	Examiner
Clément DOMBRY	Professor at Université de Franche-Comté	President
Imed KACEM	Professor at Université de Lorraine	Reviewer
Kim Thang NGUYEN	Associate professor at Université de Paris Saclay	Examiner
Jean-marc NICOD	Professor at ENSMM Besançon	Thesis Supervisor
Landy RABEHASAINA	Associate professor at Université de Franche-Comté	Co-Director
Andréa Cynthia SANTOS	Professor at Université Le Havre	Reviewer

Acknowledgements

My sincere thanks go to all my colleagues and friends for the great time and for encouraging me throughout this experience, especially the dearest Aicha, Bilal, Zeina, Chaker, Wissam, Vincent, and Jesus. I am very thankful to my supervisors and project members Jean-Marc Nicod, Landy Rabehasaina, Zeina Al Masry and Stéphane Chrétien who provided me an opportunity to join their teams and guided me in all the time of research. Finally, I would like to thank my family for always being there for me, especially my mother who always stood by my side.

Ce n'est pas la force, mais la persévérance, qui fait les grandes œuvres.

Samuel Johnson

Contents

List of Figures

List of Tables

Introduction

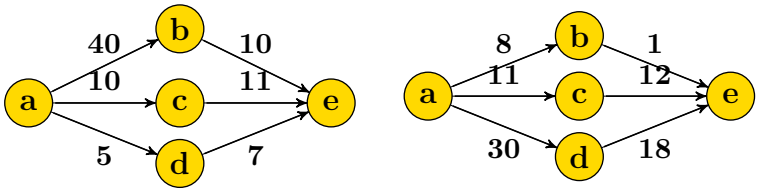
In all sectors, whether industrial or not, decisions must be made in relation to an objective that needs to be optimized. For example, in logistics, deliveries have to be made with the least amount of vehicles and the least kilometers possible; in the industry, tasks have to be processed by machines as quickly as possible in an appropriate order.

A challenge for decision making is its sustainability. What happens if the route taken by a tour carries more traffic than expected? If we had known this before, this road would have been avoided. Similarly, what if a machine is delayed in completing a task? The delay cannot be made up. Thus, it would be better to propose a good solution close to the optimal one, which remains good even if the context of the problem changes: a road has a denser traffic, the duration of a task is longer than expected. Today, given the omnipresence of optimization problems, it becomes essential to take into account the uncertain nature of the data that describes the studied problem.

Consider the example illustrated in Figure 1. A person wants to go from his house represented in node a to the train station represented in node e. He has a train to catch, and he needs to be there as fast as possible. Based on the graphs in Figure 1, this person has three possible paths: $a \to b \to e$, $a \to c \to e$, or $a \to d \to e$. Unfortunately, in his situation, uncertainty occurs in the duration of the roads. There are two possible scenarios in the street traffic state, but he needs to make a decision about what road to take. Based on Table 1, the fastest solution in the first scenario is $a \to d \to e$ with a cost of $5 + 7 = 12$. For Scenario 2, it is $a \to b \to e$, with a duration of 9 minutes. This decision is hard to take, since if the person chooses the optimal solution of one scenario but the other one occurs, then he may miss his train. Path $a \to b \to e$ takes 9 minutes in Scenario 2, but it is a very bad choice in Scenario 1: it takes 50 minutes! Contrarily, a good solution that is neither optimal is Scenario 1 nor Scenario 2 is $a \to c \to e$.

This example tells us that:

- Usually, the decision maker is put in situations where the decision has to be made before knowing the actual situation.
- If uncertainty exists, neglecting it and considering optimal solutions can have bad consequences.



(a) Scenario 1 (b) Scenario 2

Figure 1: An uncertainty situation: Decide how to go from a to e with a minimal cost when two scenarios are possible.

Possible paths	Cost in Scenario 1	Cost in Scenario 2
$a \to b \to e$	50 min	9 min
$a \to c \to e$	21 min	23 min
$a \to c \to e$	12 min	48 min

Table 1: The costs of the possible paths in both Scenarios 1 and 2 presented in Figure 1.

- A non optimal solution that behaves well in global could be better than a solution that is good in some cases, and bad in others.

In opposite to the example just given, real life problems are bigger and more complex. It is hard to compute the optimal solutions of all the possible scenarios, and considering the uncertainty becomes very challenging. Some of the challenges are given below:

1. How to exploit the uncertainty with the information at hand?
2. How to model the uncertainty in a way that is both tractable and close to reality?
3. What definition of a "good" solution that considers the uncertainty is the most reliable for the considered problem?
4. How to solve efficiently the problem that risks to be more complex when considering the uncertainty?
5. How to evaluate this solution with good metrics and in an efficient way?

The goal of this PhD thesis is first to discover and understand the different approaches to deal with the uncertainty, since it is less interesting to suppose that all the information that describes real life problems are constant and precise. The second goal is to put the approaches available in the service of discrete optimization problems, that form the majority of real life decision problems. More concretely, the thesis work was about answering the five challenges listed above.

Contributions

The contributions of this thesis start by proposing a heuristic approach for the robust counterpart of binary linear optimization problems with some assumptions linked with the exact binarity relaxation. The robust model concerned is the absolute robust definition under the ellipsoidal uncertainty set [Al Dahik 20]. The second part of contributions is the lower bound proposition for validation of heuristic approaches that solve the robust problem. This second part is composed of multiple contributions, that start with the formulation of the robust problem as a quadratic problem in order to obtain its bidual problem, and that also include the tackling of the numerical challenges of the bidual problem that turns out to be memory consuming. This has been done by a proposition of a sparse version of Pierra's algorithm. This work can be found in [Al Dahik 21]. The third contribution is the proposition of another Frank-Wolfe based algorithm for the robust `k`-median clustering problem. Finally, the thesis also contains a contribution in `k`-median clustering, by proposing a heuristic approach to solve the robust `k`-median clustering problem under ellipsoidal uncertainty, that is also based on Frank-Wolfe Algorithm.

Outline

This thesis is organized as follows. Chapter I presents the state-of-the-art of the ways to tackle uncertainty in discrete problems, namely robust optimization. Chapter II presents the chosen model in the thesis to consider uncertainty, which is the absolute robust definition under ellipsoidal uncertainty. Next, it describes the methods to solve, which are the exact method, and the heuristic approach DFW proposed to solve the robust problem, along with the corresponding numerical results. Chapter III starts with the bidualization procedure to get the problem to solve for obtaining the lower bound, then presents the methods to solve it, namely the exact method, and then the proposed method based on Pierra's algorithm. It ends with the numerical results and then some difficulties in the experiments with their interpretations. Chapter IV starts with the formulation of the `k`-median clustering, and then details the reformulation step to fall in the same form as in Chapter II. Then, it shows that the assumptions are not satisfied. Next, Chapter IV proposes a new approach to solve it heuristically and then presents the numerical results. The thesis ends with conclusions and perspectives.

Chapter I
State-of-the-art of robust optimization

This chapter introduces some of the most known approaches that exist in the state-of-the-art of robust optimization. Section I.1 gives a general overview about considering uncertainty in optimization problems. Next, Section I.2 lists some of the classical approaches that deal with uncertainty, with a focus on min-max optimization. Finally, a synthesis with a comparative table of different approaches can be found in Section I.3. In this chapter and in the whole PhD thesis, complexity theory is a prerequisite. We refer the readers to [Ausiello 12, Garey] for a global knowledge about the complexity of optimization problems.

I.1 General overview

In etymology, uncertainty refers to epistemic situations involving imperfect or unknown information. It applies to predictions of future events, to physical measurements that are already made, or to the unknown. Usually, some information about the uncertain variables are available. The variables could be random with a known distribution, or if the distribution of the variables is unknown, sometimes they belongs to a set. In other environments, the information is updated with time, and it is possible to make use of the experiences of the past to make the present decision.

Different approaches exist to tackle the uncertainty in optimization problems. The main difference between these approaches is the list of assumptions made about the knowledge of the uncertain variables. Generally, optimization problems become harder to solve when considering the uncertainty. This motivates some research directions that consider some assumptions that do not represent very well the reality, but have the advantage of making the problem relatively easy to solve. Another research direction aims to study the complexity of categories of problems, and to find if they are easy or hard.

First, let us clear up the distinction between sensitivity analysis and robust optimization. Sensitivity analysis is a post-optimization approach to deal with uncertainty [Saltelli 02]. For fixed parameters of the optimization problem, sensitivity analysis consists in solving the corresponding optimization problem, and then determining the range of variation of parameters for which the solution is still optimal. Contrarily, in the robust optimization approaches, the uncertainty is considered directly in the optimization stage.

Second, let us do a brief overview of different approaches in robust optimization. Many definitions of robustness have been proposed in the literature in the context of optimization. In a landmark paper [Ben-Tal 09], Ben-Tal *et al.* studied robust optimization in general, in the sense that the decision variables take continuous values. On the other hand, the considered case in the industrial and transporta-

tion problems is the one of discrete variables, most often so, integer or binary. The three most common definitions in the context of combinatorial optimization have been formalized in 2004 by Kouvelis and Yu in [Kouvelis]. These are absolute robust solution, robust deviation and relative robust solution. In all these cases, worst case behavior is considered. Another family of definitions are scenario dependent. In these methods, a decision is taken conditional on the current scenario and the overall optimization problem boils down to a robust two-stage problem [Ben-Tal 04]. This approach splits into the notions of K-adaptability [Hanasusanto 16], adjustable robustness [Ben-Tal 04], bulk robustness [Adjiashvili 15] and recoverable robustness [Liebchen 09]. In the case where the data can be considered as governed by a certain probability distribution with unknown parameters, distributionally robust optimization [Rahimian 19] is also an interesting approach. It consists in choosing the distribution that is most suitable given a robustness criterion. Yet another approach is the notion of almost robust solution [Baron 19] that is feasible under most of the realizations and that can use full, partial or no probabilistic information about the uncertain data. Let us also mention that other alternative generic approaches have also been proposed in the literature: Buhmann *et al.* consider in [Buhmann 18] a near-optimum solution for several scenarios. Another way to tackle uncertainty that is different from robust optimization is the online optimization [Hazan 16], where decisions are made iteratively, and at each iteration, the problem inputs are unknown, but the decision maker learns from the previous configuration before making his decision. After a decision is made, it is then assessed against the optimal one. Finally, let us add that uncertainty theory was used in another line of work as in Liu *et al.* in [Liu 09], for instance. This theory has also been implemented by Gao *et al.* in [Gao 11] in order to give what they call an uncertainty distribution in the case of the shortest path problem.

Finally, it is important to know that some problems are too complex to fit in any theory, especially in real life problems, for example for the problem of power provisioning in data centers, that depends on the weather condition and many other factors [Pierson 19]. This challenge has opened to a big project on hold, in order to manage the uncertainty in such a complex system [1] [Haddad 21]. Another example that needs particular tools is the one of scheduling, which led to an information-based decision tree model in [Portoleau 20]. A third example is a real life challenge of metal coils assignment studied in [Omri 20], where metal properties are collected with uncertainty. This uncertainty affects the detection of metal coils suitability to the production process. A last example is a challenge in the traveling salesman problem that has been tackled in [Toklu 17], where the

[1] https://anr.fr/Project-ANR-19-CE25-0016

classical robust optimization does not represent well the real uncertainty, and thus it has been considered that it is more accurate to create uncertainty profiles to a activated at a certain period of time, dynamically.

The next section gives formal definitions of most of the approaches listed above, that are: min-max robust optimization, robust two-stage optimization, distributionally robust optimization, online optimization, and an approach under the framework of uncertainty theory.

I.2 Approaches to tackle uncertainty in optimization problems

This section maps some of the approaches to deal with the uncertainty: min-max robust optimization, robust two-stage optimization, distributionally robust optimization, sensitivity analysis, online optimization, the approach under the framework of uncertainty theory, and some problem specific approaches. It is important to point out the confusion for the keyword robust optimization: in general reviews, robust optimization can refer to any approach that deals with uncertainty. On the other side, and in particular, robust optimization sometimes refers to the min-max approaches, that are the min-max robust optimization described in Section I.2.1 and the robust two-stage optimization described in Section I.2.2.

I.2.1 Min-max robust optimization

The approach described in this section supposes that some information about the uncertainty is at hand, which can be represented in an uncertainty set, and hedges against the worst behavior in this uncertainty set. This section is extensively developed, since it is studied in this PhD thesis. The motivation of this choice is explained in Chapter II.

Consider a general optimization problem

$$\min_{x \in X} f(x), \tag{I.1}$$

where f is the objective function to be minimized, and X is the feasible set. Suppose that d is the data describing the problem definition. Then, f and X depend on d, and the problem is equally written as

$$\min_{x \in X_d} f(x, d), \tag{I.2}$$

where X_d is the feasible set under the realization d. When we suppose that the data is certain and constant, we call the problem "deterministic".

If d is uncertain, and if the information at hand about d allows to define an uncertainty set U such as $d \in U$, then three common definitions of robust solutions were studied in [Soyster 73], then in [Kouvelis], which are the absolute robust decision, the robust deviation decision, and the relative robust decision. Here, the worst case behavior is considered .This family of definitions are called min-max optimization problems, and they are stated in Section I.2.1.b. The forms of the uncertainty set U are described in Section I.2.1.c. In order to illustrate the definitions of min-max decisions, as well as the uncertainty sets, a pedagogical example is considered, and it is described in Section I.2.1.a.

I.2.1.a Description of the pedagogical example

Section I.2.1 is supported with a simple example that follows the different sections, in order to accompany the reader for easier understanding of the different definitions. Figure I.1 represents the considered example. The goal is to go from point a to point b, with the least cost possible, and the possible paths are : abe, ace and ade. The edges ab, ac, ad, be, ce and de are weighted with the costs of these edges. The cost of path abe is the sum of the weights of the edges ab and be. The considered minimization problem can be written as the following problem

$$\min_{x \in X} c^T x,$$

where $X = \{[1,0,0,1,0,0], [0,1,0,0,1,0], [0,0,1,0,0,1]\}$ is a binary representation of the paths abe, ace and ade. If we note the edges as the following

$$\begin{aligned} e_1 &= ab, \\ e_2 &= ac, \\ e_3 &= ad, \\ e_4 &= be, \\ e_5 &= ce, \\ e_6 &= de, \end{aligned}$$

then a path $x \in X$ is a vector with 6 elements, with x_i equals 1 if the path x contains the edge e_i, and 0 otherwise. For example, abe is represented by $[1,0,0,1,0,0]$, since it contains $e_1 = ab$ and $e_4 = be$. The cost of a path x equals $c^T x$, where $c = [c_1, \ldots, c_6]$ represents the costs of the edges $\{e_1, \ldots, e_6\}$.

I.2.1.b Absolute robuste decision, robust deviation decision and relative robust decision

The following states the formal definitions of the absolute robust decision, the robust deviation decision, and the relative robust decision.

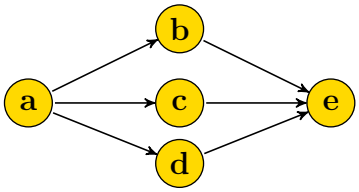


Figure I.1: Pedagogical example: a graph with three possible paths from node a to node e.

I.2.1.b.1 Absolute robust decision: The first called absolute robust decision is defined as the solution of the problem

$$z_A = \min_{x \in \cap_{d \in U} X_d} \max_{d \in U} f(x, d). \tag{I.3}$$

The absolute robust decision is the result of a comparison between the worst cases of all the solutions: it is the best solution in its worst case in the uncertainty set U. This solution must be feasible in all the scenarios of $d \in U$. If the user chooses the absolute robust solution, despite whatever scenario he falls in, his solution is better than the worst cases of the other solutions. This definition is criticized for being pessimistic, since it protects against the worst case, no matter how likely or unlikely it is to occur. Nevertheless, in some applications, it is important to anticipate the worst case, such as nuclear accidents or public health [Aissi 09].

In the case of the example in I.2.1.a, if the uncertainty occurs only in the costs of the edges, such that the cost vector c belongs to an uncertainty set U, then

$$z_A = \min_{x \in X} \max_{c \in U} c^T x. \tag{I.4}$$

I.2.1.b.2 Robust deviation decision: Next, the robust deviation decision is defined as

$$z_D = \min_{x \in \cap_{d \in U} X_d} \max_{d \in U} \Big(f(x, d) - f(x_d^*, d) \Big), \tag{I.5}$$

where x_d^* is the optimal solution in the scenario of d. This definition is less conservative, since it protects against the worst deviation from optimality: the robust deviation decision is the closest to the optimal in its worst case. For a decision x, in a certain scenario d, the difference

$$R = f(x, d) - f(x_d^*, d)$$

is also called regret, and thus this robust decision is also called min-max regret, and it minimizes the worst regret [Kacem 19, Aissi 05, Aissi 07].

In the case of the example in I.2.1.a, we have

$$z_D = \min_{x \in X} \max_{c \in U} \Big(c^T x - c^T x_c^* \Big). \tag{I.6}$$

I.2.1.b.3 Relative robust decision: Another version of the previous robust decision is with a relative deviation [Coco 14]. This yields to the third definition of a robust decision, the one called relative robust decision. It is defined as

$$z_R = \min_{x \in \cap_{d \in U} X_d} \max_{d \in U} \frac{f(x,d) - f(x_d^*, d)}{f(x_d^*, d)}. \tag{I.7}$$

In the case of the example in I.2.1.a, we have

$$z_R = \min_{x \in X} \max_{c \in U} \frac{c^T x - c^T x_c^*}{c^T x_c^*}. \tag{I.8}$$

The robust deviation decision and the relative robust decisions are adapted to the applications in highly competitive market environments, where the performance needs to be satisfactory in any realization of the uncertain variable, for example in the investment management.

The remainder of Section I.2.1 discusses in greater detail the absolute robust optimization (Definition (I.3)), and considers a particular class of problems in the general form (Problem (I.1)), where $f(x,c) = c^T x$, and $X \subseteq \{0,1\}^m$. We also suppose that the uncertainty occurs only in the cost vector $c \in \mathbb{R}^m$. Thus, we consider the robust counterpart of problems following the form:

$$\min_{x \in X} c^T x, \tag{I.9}$$

where $X \subseteq \{0,1\}^m$ is the set of the feasible solutions, and where the cost vector $c \in \mathbb{R}^m$ is subject to uncertainty, i.e., it has more than one possible realization. Let U be an uncertainty set included in $\mathbb{R}^m$ ($U \subseteq \mathbb{R}^m$).

The absolute robust counterpart of Problem (I.9) (Problem (I.3)) is then defined as:

$$\min_{x \in X} \max_{c \in U} c^T x. \tag{I.10}$$

There are multiple types of the uncertainty set U, that will be described in the following.

I.2.1.c Uncertainty sets

The choice of the uncertainty set is critical for the complexity of the robust problem. It is chosen depending on many reasons, especially the information available about the uncertain variables, and the level of difficulty of solving the robust problem. The three most common uncertainty sets are the discrete set, the interval

e_i	$c_i^{(1)}$	$c_i^{(2)}$	$c_i^{(3)}$
$e_1 = ab$	2	4	1
$e_2 = ac$	7	2	8
$e_3 = ad$	3	4	2
$e_4 = be$	4	4	3
$e_5 = ce$	2	1	4
$e_6 = de$	2	3	4

Table I.1: The values of the uncertain costs in the different scenarios.

x	$c^{(1)\,T}x$	$c^{(2)\,T}x$	$c^{(3)\,T}x$	$\max_{c\in U} c^T x$	$\max_{c\in U} c^T x - c^T x_c^*$
abe	6	8	4	8	5
ace	9	3	12	12	8
ade	5	7	6	7	4
min	$\min_{x\in X} c^{(1)\,T}x = 5$	$\min_{x\in X} c^{(2)\,T}x = 3$	$\min_{x\in X} c^{(3)\,T}x = 4$	$z_A = 7$	$z_D = 4$

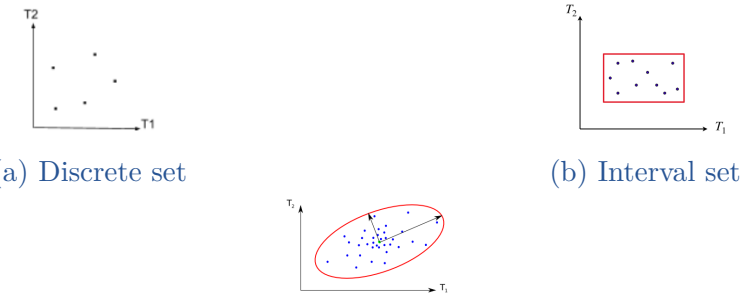
Table I.2: Solutions: the optimal solution in every scenario, the absolute robust decision z_A , and the robust deviation decision z_D in the case of a discrete uncertainty set.

set, and the ellipsoidal uncertainty set. They are illustrated in Figure I.2, that is a two-dimensional illustration of the discrete set in I.2a, the interval set in I.2b, and the ellipsoidal set in I.2b. In this figure, the discrete set is represented by a set of finite points, the interval set is represented by a rectangle that delimits the two intervals on the two axes that represent the uncertain elements. The ellipsoid is represented by an ellipsoid with inside some points that are modelled by a bi-normal distribution: the points are grouped around the mean value, and they spread more and more when they get far from the mean.

- Discrete set

This uncertainty set is considered in the cases where the scenarios are explicitly given, i.e., the possible cases that form the uncertainty set. Usually, it is obtained out of data from the past. U has the form $U = \{c_1, \ldots, c_n\}$, n being the number of scenarios. This set is not considered much in practice because it leads to difficult problems due to the discrete aspect of this set.

In the case of the example in I.2.1.a, suppose that the cost c is uncertain, and it belongs to a discrete set of 3 scenarios $c \in U = \{c^{(1)}, c^{(2)}, c^{(3)}\}$. The values are given in Table I.1. Next, Table I.2 aims to show how the definitions of the absolute decision z_A and the deviation decision z_D are applied on an example for



(a) Discrete set

(b) Interval set

(c) Ellipsoidal set

Figure I.2: A two-dimensional representation of the discrete (a), interval (b) and ellipsoidal sets (c).

discrete sets. This helps to understand the min-max optimization in general. For a scenario s, $s \in \{1, 2, 3\}$, the optimal solution is $\min_{x \in X} c^{(s)\,T} x$. For example, in scenario 1, the costs are 6, 9 and 5. The lowest is 5: this is the optimal solution in scenario 1. Next, to find the absolute robust decision z_A, two steps are needed. The first step is to find the worst cost among the 3 scenarios for every solution x: it is represented in the column $\max_{c \in U} c^T x$ of Table I.2. The second step is to find the best of these worst cases to get z_A. On the other side, to compute the robust deviation decision z_D, we make use of the optimal solutions in each scenario to compute the deviations from optimality, and we proceed in the same way as for z_A: we first compute the worst deviation among the scenarios for the paths abe, ace, and ade. Finally, we choose the least value among the worst deviations to get the solution z_D. As one can realize, the deviation decision requires the computation of the optimal solutions of all the scenarios, whereas it is not the case for the absolute decision.

- Interval set

This uncertainty set is considered when every entry of the uncertain cost vector c varies within an interval, i.e., between a lower bound and an upper bound. U can be written as $U = \Pi_{i=1}^{m} [\underline{c_i}; \overline{c_i}]$, where m is the dimension of the uncertain variables, $\underline{c_i}$ (resp. $\overline{c_i}$) is the proposed lower (resp. upper) bound of c_i, $i \in \{1, \ldots, m\}$. Moreover, in this development, it is assumed that $U \subseteq \mathbb{R}_+$. The worst case for this uncertainty set is when all the cost elements are on their upper bound. This uncertainty set can make the robust counterpart tractable, and for many problems, the complexity of the robust version is the same as the deterministic problem. In our case Problem (I.10) becomes

$$\min_{x \in X} \max_{c \in U} c^T x = \min_{x \in X} \max_{\substack{\underline{c_i} \leq c_i \leq \overline{c_i} \\ i \in \{1, \ldots, m\}}} c^T x = \min_{x \in X} \overline{c}^T x. \tag{I.11}$$

Thus, the robust counterpart of Problem (I.9) in case of the interval uncertainty set has the same complexity of the deterministic problem (I.9) (i.e. without uncertainty). Nevertheless, this uncertainty definition is too conservative and pessimistic, since the considered case is when all the elements are in their worst case, which is not likely to occur.

A way to avoid the over pessimism of interval uncertainty is to believe that in real life, not all coefficients are uncertain at the same time. This motivates the budgeted uncertainty, also called Γ-uncertainty : a particular case of min-max optimization with interval uncertainty sets. Budgeted uncertainty has been proposed by Bertsimas and Sim in [Bertsimas 03], and it consists in supposing that a fixed number Γ of elements of the uncertain vector are uncertain, and the others are constant.

If we recall the interval uncertainty set

$$U = \Pi_{i=1}^{m}[\underline{c_i}; \overline{c_i}],$$

then, the uncertainty set proposed here is

$$\begin{aligned} U = \Big\{ c = \{c_1, \ldots, c_m\} \in \mathbb{R}^m; \\ \underline{c_i} \leq c_i \leq \overline{c_i} \quad \forall i \in I, \\ c_i = \frac{\underline{c_i} + \overline{c_i}}{2} \quad \forall i \notin I, \\ I \subseteq \{1, \cdots, m\}, |I| \leq \Gamma \Big\}. \end{aligned}$$

Or, in another writing

$$\bigcup_{\substack{|I| \leq \Gamma \\ I \subseteq \{1, \cdots, m\}}} \Big\{ \Pi_{i \in I}[\underline{c_i}; \overline{c_i}] \times \Pi_{i \in I}\{\frac{\underline{c_i} + \overline{c_i}}{2}\} \Big\}.$$

This uncertainty set contains vectors c, such that not all of their elements are uncertain: at most Γ elements are uncertain in an interval $[\underline{c_i}; \overline{c_i}]$, and the others have a certain value that equals $\frac{\underline{c_i} + \overline{c_i}}{2}$. For $\Gamma = m$, the budgeted uncertainty set becomes the interval set, and for $\Gamma = 0$, the values of c are certain, and thus we find the deterministic problem. Bertsimas and Sim in [Bertsimas 03] show that the robust counterpart (I.10) with the budgeted uncertainty set can be solved by solving at most $m + 1$ instances of the deterministic problem (I.9). To conclude, the budgeted uncertainty is a way to undirectly reduce the pessimism of the interval uncertainty set, while keeping the complexity of the deterministic problem [Poss 13, Bougeret 19].

- Ellipsoidal set

A third uncertainty set that takes into account the correlation between the elements of the vector c is the ellipsoidal uncertainty set. Imagining that we have a discrete set of observations for the vector c (like for the discrete uncertainty set), one way to reduce the combinatorial aspect of the discrete set is to consider the following ellipsoid as the uncertainty set

$$U = \{c \in \mathbb{R}^m; (c-\mu)^T \Sigma^{-1} (c-\mu) \leq \Omega^2\},$$

where $\mu \in \mathbb{R}^m$ is the sample mean, $\Sigma \in \mathbb{R}^{m \times m}$ is the sample covariance matrix and $\Omega > 0$. This proposition comes from the assumption that c has a multinormal distribution with expectation μ and covariance matrix Σ. U is then a confidence set for c, which means that c belongs to U with a certain probability that is controlled by the scalar Ω. As already mentioned, the advantage of this uncertainty set is that it takes into consideration the correlation between the elements of the vector c, which is close to reality, since for example, a road with a traffic jam probably causes congestion in other roads. The second advantage is that it allows the user to control the level of risk that he is ready to take in order to have a good cost. This is possible by the choice of Ω, since with a small Ω, the user can eliminate less likely cases when finding the best solution in its worst case. The third advantage is that it leads to a smooth form for the min-max formulation. Indeed, with the ellipsoidal uncertainty set, Problem (I.10) becomes:

$$\min_{x \in X} \mu^T x + \Omega \sqrt{x^T \Sigma x}. \tag{I.12}$$

The proof is developed in Section II.2. In another interpretation, Problem (I.12) is a minimization of a weighted sum of the mean $\mu^T x$ and the risk $\sqrt{x^T \Sigma x}$ of x. It is very known in portfolio optimization, and it is called mean-risk optimization [Markowitz 52].

Unfortunately, min-max optimization problems under general ellipsoidal uncertainty are NP-hard. For exact and heuristic methods to solve Problem (I.12), see [Buchheim 18a] and [Al Dahik 20].

On the other hand, in the special case of the uncorrelated uncertainty set, i.e., when Σ is a diagonal matrix $\Sigma = \text{diag}(d_1, \ldots, d_m)$, with $d = \{d_1, \ldots, d_m\}$ representing the variance of c, Problem (I.12) becomes

$$\min_{x \in X} \mu^T x + \Omega \sqrt{d^T x}. \tag{I.13}$$

It is not known yet if this problem has the same complexity of the deterministic problem. Nevertheless, Bertsimas and Sim in [Bertsimas 04b] show that if $d_1 = d_2 = \ldots = d_m$, the robust problem has the same complexity as the deterministic problem. Moreover, they propose in the same reference a heuristic algorithm to solve the problem with uncorrelated but not identically distributed data.

Other uncertainty sets were also defined and studied, for example polytopic uncertainty sets, uncertainty sets under general norms [Bertsimas 04a], uncertainty sets modeled by multiple knapsack constraints ([Poss 18]), etc. For a more advanced study about the different uncertainty definitions, see [Li 11] and [Buchheim 18b].

I.2.2 Robust two-stage optimization

Since the robust optimization approaches proposed in the previous section are considered as too conservative, and the min-max optimization implies that the decision is made before the uncertain variables are revealed, an other approach named min-max-min optimization or robust two-stage optimization has been proposed by Buchheim and Kurtz in [Buchheim 17]. This approach splits into the notions of K-adaptability [Hanasusanto 16], adjustable robustness [Ben-Tal 04], bulk robustness [Adjiashvili 15] and recoverable robustness [Liebchen 09]. The robust two-stage problem is the one where some decisions can be made after the scenario is known. The robust two-stage problem is defined as

$$\min_{x\in X} \max_{\xi\in U} \min_{\substack{y\in Y \\ (x,y)\in Z_\xi}} f_\xi(x,y),$$

where $x \in X$ are the first-stage decisions, $y \in Y$ are the second-stage decisions. f_ξ is the objective function that depends on the uncertain variable ξ that belongs to an uncertainty set U, and Z_ξ is the feasibility set of (x, y) under the scenario ξ.

This approach is adapted for problems that have a possibility of recourse, such as for parcel delivery companies, who do not reschedule their delivery tours from scratch every time. Instead, drivers are trained for a small part of route plans that are executed in case of road closures and congestion. This example is taken from [Arslan 20].

In general, these problems are extremely hard to solve. An example of a method to solve such problems can be found in [Hanasusanto 15].

I.2.3 Distributionally robust optimization

This section presents the distributionally robust optimization, a middle ground between stochastic optimization and absolute robust optimization [Rahimian 19, Goh 10]. Let us first introduce the stochastic optimization.

Consider again a general problem in the form (I.1), that is we recall

$$\min_{x\in X} f(x). \tag{I.14}$$

If the objective function f depends on a variable d that represents a random variable with a joint cumulative distribution F_d, the stochastic optimization problem is the problem of minimizing the expected value of this function. It consists in

solving the following problem

$$\min_{x \in X} \mathbb{E}_{F_d}[f(x, d)].$$

In the cases where it is not crucial to consider the worst case behavior, stochastic optimization seems to be a good choice for considering the uncertainty. Nevertheless, it considers that a full knowledge is available about this uncertainty, by the existence of a known distribution of the uncertain variable d. In the absolute robust optimization defined by Problem I.3, we suppose that the distribution of the uncertain variable d is unknown, and we minimize the worst case in an uncertainty set, no matter how likely or unlikely it is to occur. A middle ground solution is to consider having a limited information about the distribution, and to have a robustness in the proposed solution. This introduces to the distributionally robust optimization that is described in the following. In the case where the data can be considered as governed by a certain probability distribution with unknown parameters, distributionally robust optimization consists in choosing the distribution that is most suitable given a robustness criterion. It corresponds to solving the following problem

$$\min_{x \in X} \max_{F_d \in \mathcal{D}} \mathbb{E}_{F_d}[f(x, d)],$$

where $\mathcal{D}$ is a set of distributions that supposedly includes the true distribution of d. In this definition, we choose the solution that is the best for the worst possible expected value among the ones corresponding to the distributions of the set $\mathcal{D}$.

To know more about the choice of the set of distributions and an application to portfolio optimization, see [Delage 10].

I.2.4 Online optimization

Another way to tackle uncertainty that is different from robust optimization is the online optimization introduced in [Zinkevich 03], also considered as an online learning, since the learner makes the decisions iteratively, and at each new iteration, the problem inputs are unknown, but the learner has access to the previous inputs, and thus learns from these inputs to decide [Shalev-Shwartz 07]. After every decision is made, a metric to evaluate the performance of the online player is called regret, which is the gain of the best decision minus the gain of the learner. Different categories in online optimization exist, depending on the information available for the learner. An online optimization problem can be seen as a game between the learner and the environment, that are named in the context of a game as the player and the adversary, respectively. It is described as the following: at every time step t, the player chooses a decision x_t in the set of decisions X, the

adversary chooses f_t, the objective function at time step t. The player suffers his loss and observes a feedback, the feedback being, whether a partial information $f_t(x_t)$ or a full information f_t. The goal of the player at the last iteration T is to minimize the regret R_T, which is defined as follows

$$R_T = \Sigma_{t=1}^{T} f_t(x_t) - \min_{x \in X} \Sigma_{t=1}^{T} f_t(x).$$

More details about this problem with an application on the online shortest path problem can be found in [Abernethy 09].

I.2.5 Approach under the framework of uncertainty theory

Another line of work has been discovered under the framework of uncertainty theory. This theory has been founded in 2007 by Liu *et al.* in [Liu 09], and it leads to define an uncertain variable, in analogy to the definition of a random variable with probability theory. Uncertainty theory states that a random variable is an uncertain variable, but the opposite is not true. Thus, under the framework of this theory, and if we suppose that we have an uncertain variable, it is possible to compute an uncertainty distribution, that allows to compute an optimal solution under a confidence level, that is a way to tackle the uncertainty differently than the other approaches we have seen so far. This theory has been implemented by Gao *et al.* in [Gao 11] in order to give what they call an uncertainty distribution in the case of the shortest path problem.

I.3 Synthesis

After discovering different approaches for dealing with uncertainty, one can deduce that some approaches and elements are comparable. Indeed, Robust min-max optimization described in Section I.2.1 can be compared to other approaches, following defined criteria. More precisely, five comparisons have been deduced in Table I.3: the ones where we compare robust min-max optimization with (1) sensitivity analysis, (2) stochastic optimization, (3) robust two-stage optimization, (4) online optimization, and (5) the approach based on uncertainty theory. For example, comparing robust min-max optimization with sensitivity analysis can be done following two criteria: the first criteria is the type of action: robust min-max optimization considers the uncertainty, whereas sensitivity analysis analyzes the uncertainty, etc. It is noted that not all the approaches are comparable, and for the ones that are comparable, the comparison is not done on the same level and the same criteria of comparison for all of them.

The next chapters are the contributions of this PhD thesis in the robust optimization domain. All the study is based on the absolute min-max optimization under ellipsoidal uncertainty sets, that has been detailed previously in Section I.2.1. The motivation of this choice is detailed below in Chapter II.

Comparison criteria	Approach 1	Approach 2
	Robust min-max optimization	Sensitivity analysis
Type of action	Consider uncertainty	Analyze uncertainty
Step of action	In optimization	Post-optimization
	Robust min-max optimization	Stochastic optimization
Considered case	Worst case	Mean value
Model assumption	No distribution	Probability distribution
Type of application	For extreme events	For regular events
Complexity	Hard	Easy
	Robust min-max optimization	Robust two-stage optimization
Type of action	Without recourse	With recourse
Complexity	Hard	Harder
	Robust min-max optimization	Online optimization
Type of action	Considers uncertainty once and for all	Considers progressively the uncertainty
	Robust min-max optimization	Uncertainty theory
Model assumption	No distribution	Uncertainty distribution

Table I.3: Comparison table between robust min-max optimization and other approaches.

Chapter II

A heuristic approach for robust discrete optimization: first example on the robust shortest path problem

This chapter deals with the robust counterpart of binary linear programming problems in the case of correlated ellipsoidal uncertainty set. We show that this problem is hard, and despite the existence of exact methods, they become costly when the problem is big. Thus, the first contribution of this thesis is to propose a heuristic algorithm based on Frank-Wolfe's algorithm for the robust counterpart of problems with exact relaxation such as the shortest path problem. For this, an explanation of how the classical Frank-Wolfe Algorithm works is first provided, and the mechanism of the proposed Algorithm is then revealed. The approach is validated on the example of the robust shortest path problem for problems from small to medium size in comparison with an exact solver based on branch-and-bound methods. Finally, the performance of the proposed algorithm is shown by numerical comparisons.

II.1 Motivation and context

A complete formal definition of the chosen model and the considered problem, as well as the motivations of every choice can be found fully in Section II.2. Nevertheless, this section aims to group all the elements of the model and motivations in one paragraph, for the sake of easier accessibility for the reader.

As already motivated in the introduction of this PhD thesis, it is important to consider the uncertainty in optimization problems, since neglecting it could have bad consequences in the output of the problem: an optimal solution could be far from optimal if the input of the problem changes. Since considering the uncertainty makes the problems harder to solve in general, we consider to deal with the uncertainty that is encountered in binary linear problems. We also suppose that the uncertainty occurs only in the objective function.

A second choice to make is the model of uncertainty. We choose the absolute robust decision, because we have strong guarantees about the robust solution: with this definition, the proposed decision is always the best in its worst behavior.

Next, since this robust decision is criticized for being too pessimistic, we choose to accompany it with the model of ellipsoidal set as the uncertainty set, since it allows to control the pessimism. This uncertainty set has other advantages, such as considering the correlations of the inputs of the problem, and the fact that it gives a smooth form of the min-max problem as a minimization problem.

For the exact method in Section II.3, no additional assumptions are considered, whereas, for the proposed heuristic approach in Section II.4, three assumptions are needed, which are linked with the exact integrality relaxation.

All the details can be found in the following.

II.2 Problem formulation

We consider binary linear optimization problems of the form:

$$\min_{x \in X} c^T x, \tag{II.1}$$

where $X = \{x \in \{0,1\}^m; Ax = b\}$ is the set of the feasible solutions, and where the cost vector $c \in \mathbb{R}^m$ is subject to uncertainty, i.e., it has more than one possible realization. Let U be an uncertainty set included in $\mathbb{R}^m$ ($U \subseteq \mathbb{R}^m$). The absolute robust counterpart of Problem (II.1) is then defined as:

$$\min_{x \in X} \max_{c \in U} c^T x. \tag{II.2}$$

Problem (II.2) is motivated as follows. If we suppose that c takes its values in U, then solving Problem (II.2) amounts to finding the optimal solution in X corresponding to the worst possible cost. The advantage of this solution is that it guarantees that no matter what scenario occurs in U, the decision maker is sure that his decision costs him less than the worst cases of every other decision.

If we consider that the cost coefficient vector c has a multinormal distribution with expectation $\mu \in \mathbb{R}^m$ and covariance matrix $\Sigma \in \mathbb{R}^{m \times m}$ (see e.g. [Gut , Chapter 5]) , an interesting uncertainty set to consider is the following ellipsoid:

$$U = \{c \in \mathbb{R}^m; (c - \mu)^T \Sigma^{-1} (c - \mu) \leq \Omega^2\}. \tag{II.3}$$

Indeed, U is a confidence set for c, i.e., c belongs to U with probability $1-\alpha \in [0,1]$, where $\Omega > 0$ is written as $\Omega_\alpha = \chi^2_m(1-\alpha)$ and $\chi^2_m(1-\alpha)$ refers to the quantile function for probability $1-\alpha$ of the chi-squared distribution with m degrees of freedom . The covariance matrix Σ is positive semi-definite, but it is both useful and reasonable to suppose that Σ is symmetric positive definite and not only symmetric positive semi-definite. In general, the covariance matrix is positive definite unless one variable is an exact linear function of the others. Thus, it is sometimes common to take this assumption about the covariance matrix like in [Ilyina 17], since it is useful for its properties that we see in this manuscript in many places.

In that case, the parameter Ω describes the level of confidence in solving the corresponding problem (II.2), i.e., the risk the user is willing to take. If $\Omega = \Omega_\alpha$ is small, i.e., α is close to 1, the user is willing to accept the risk to obtain a better solution , while a user who chooses a bigger value for Ω prefers to be secured in

more cases, even if these cases are less likely to occur. This implies a big advantage of the ellipsoidal uncertainty set, especially in opposite of the interval uncertainty set, since it reduces the pessimism of the absolute robust definition, due to the possibility to eliminate the cases that are less likely to occur, by the choice of Ω. A second advantage is the consideration of the correlation between the elements of the vector c, whereas, it is not the case for the interval uncertainty set, defined in Section I.2.1.c. A third advantage of the ellipsoidal uncertainty set is expressed in the following. If U is defined by the set (II.3), then Problem (II.2) can be rewritten as a non-linear optimization problem:

$$\min_{x \in X} \mu^T x + \Omega \sqrt{x^T \Sigma x}. \tag{II.4}$$

Proof. The reformulation (II.4) is obtained the following way. First, for a fixed $x \in X$, the solution of the problem

$$\max_{c \in U} c^T x = \max_{c \in \mathbb{R}^m; (c-\mu)^T \Sigma^{-1} (c-\mu) \leq \Omega^2} c^T x \tag{II.5}$$

can be given explicitly, by defining $z = (\Sigma^{-1})^{\frac{1}{2}}(c - \mu) \iff c = \Sigma^{\frac{1}{2}} z + \mu$. Here, we need to mention that Σ is symmetric and positive definite, then it is invertible and its inverse matrix Σ^{-1} is symmetric and positive definite. Then it is possible to write $\Sigma^{-1} = (\Sigma^{-1})^{\frac{1}{2}}(\Sigma^{-1})^{\frac{1}{2}}$, with $(\Sigma^{-1})^{\frac{1}{2}}$ symmetric.Thus, Problem (II.5) is written in the following way

$$\begin{aligned}
&\max_{z \in \mathbb{R}^m; z^T z \leq \Omega^2} (\Sigma^{\frac{1}{2}} z + \mu)^T x \\
&= \max_{z \in \mathbb{R}^m; z^T z \leq \Omega^2} \mu^T x + (\Sigma^{\frac{1}{2}} z)^T x \\
&= \mu^T x + \max_{\|z\| \leq \Omega} (\Sigma^{\frac{1}{2}} x)^T z \\
&= \mu^T x + (\Sigma^{\frac{1}{2}} x)^T \frac{\Sigma^{\frac{1}{2}} x}{\|\Sigma^{\frac{1}{2}} x\|} \Omega = \mu^T x + \Omega \sqrt{x^T \Sigma x}.
\end{aligned}$$

□

Thus, we see that in the special case of the ellipsoidal uncertainty set, finding a robust solution is reduced to solving a deterministic optimization problem. In addition, one could interpret (II.6) as a weighted sum of the mean $\mu^T x$ and the risk $\sqrt{x^T \Sigma x}$, which is called a mean-risk optimization problem [Markowitz 52].

Note that we are able to write the objective function as $\mu^T x + \sqrt{x^T \Sigma x}$, by replacing Σ by $\Omega^2 \Sigma$, in order to simplify its expression. Then along this work, we would rather use the following form

$$\min_{x \in X} \mu^T x + \sqrt{x^T \Sigma x}. \tag{II.6}$$

This problem is a non-linear non-convex problem which makes it challenging to find an appropriate method to solve it. In the next section, we show an exact method to compute an optimal solution.

II.3 Method for computing an optimal solution

There exists another formulation of problem (II.6) that permits to solve it using existing algorithms which can be found in [Ilyina 17]. Problem (II.6) can be written as a Binary Second Order Cone Programming problem (BSOCP). Since Σ is positive symmetric definite, then we can write $\Sigma = \Sigma^{\frac{1}{2}}\Sigma^{\frac{1}{2}}$, with $\Sigma^{\frac{1}{2}}$ symmetric. Then we obtain:

$$\begin{aligned} & \min \mu^T x + \sqrt{x^T \Sigma x} \\ & \text{s.t. } x \in X \\ \iff \quad & \min \mu^T x + \sqrt{x^T \Sigma^{\frac{1}{2}} (\Sigma^{\frac{1}{2}})^T x} \\ & \text{s.t. } x \in X. \end{aligned} \tag{II.7}$$

With a change of variable and after adding a variable that transforms the objective function to a linear one, Problem (II.7) becomes:

$$\begin{aligned} & \min \mu^T x + z \\ & \text{s.t. } \|y\|_2 \leq z \\ & y = (\Sigma^{\frac{1}{2}})^T x \\ & x \in X, y \in \mathbb{R}^m, z \in \mathbb{R}_+. \end{aligned}$$

Finally, noticing that a cone constraint is revealed, we conclude that (II.7) is equivalent to:

$$\begin{aligned} & \min \mu^T x + z \\ & \text{s.t. } (y, z)^T \in K_{m+1} \\ & y = (\Sigma^{\frac{1}{2}})^T x \\ & x \in X, y \in \mathbb{R}^m, z \in \mathbb{R}_+, \end{aligned} \tag{II.8}$$

with $K_{m+1} = \left\{ x \in \mathbb{R}^{m+1}; \|(x_1, \ldots, x_m)^T\|_2 \leq x_{m+1} \right\}$ being a second order cone. The same problem without the integrality condition corresponds to a Second Order Cone Programming problem (SOCP) and can be solved in a polynomial time [Alizadeh 03, Section 8]. Thus, we easily obtain a lower bound for a branch-and-bound

method to get an optimal solution. There exists a BSOCP solver in CPLEX [Manual 87] that gives an optimal solution of the addressed problem using the formulation (II.8). But we mention that for problems of large size, the processing time of branch-and-bound methods may become considerable, since the size of the tree may grow exponentially. Thus proposing a heuristic algorithm seems mandatory. Such algorithm is presented in the following section.

II.4 Scalable suggested heuristic algorithm

The proposed heuristic algorithm is a variant of the Frank-Wolfe algorithm [Frank 56]. It requires some specific assumptions which are listed first. Then we recall the classical Frank-Wolfe algorithm. Finally, we describe the algorithm in detail.

II.4.1 Assumptions

From now on, we consider the robust counterparts (II.6) of problems in the form (II.1) that verify the following assumptions:

(A1) For any real-valued vector a (not necessarily with positive entries), there exists an efficient algorithm to solve $\min_{x\in X} a^T x$,

(A2) For any real-valued vector a, there exists a solution for $\min_{x\in \mathrm{Conv}(X)} a^T x$ that belongs to X, where $\mathrm{Conv}(X) \subset \mathbb{R}^m$ is the convex hull of X,

(A3) The vector with zeros in all entries $0_{\mathbb{R}^m}$ does not belong to X.

The robust shortest path problem is an example for which these assumptions are verified: polynomial time algorithms exist to optimally solve the deterministic shortest path problem with real valued costs, hence **(A1)** is satisfied. **(A2)** is verified since the incidence matrix of the graph in that problem is totally unimodular. The vector $0_{\mathbb{R}^m}$ is never a path, hence **(A3)** is satisfied. Another example is the workforce planning problem(see [Lee 04, Section 0.8]). For more details about totally unimodular matrices, see [Lee 04].

II.4.2 The classical Frank-Wolfe algorithm

This section aims to describe the Frank-Wolfe algorithm to minimize convex functions with convex constraints. This algorithm is adapted in the following section (Section II.4.3) to solve heuristically our robust problem.

Let f be a real valued, convex and continuously differentiable function defined on a compact convex D. We consider in this section general constrained convex optimization problems of the form

$$\min_{x \in D} f(x). \tag{II.9}$$

For such optimization problems, one of the simplest and earliest known iterative optimizers is given by the Frank-Wolfe method [Frank 56], also known as *conditional gradient method*

This algorithm is described in Algorithm 1.

Algorithm 1 Frank-Wolfe 1956 [Frank 56]

Let $x^{(0)} \in D$
for $k = 0$ to K **do**
  compute $s^{(k)} \leftarrow \underset{s \in D}{\operatorname{argmin}} \nabla f(x^{(k)})^T s$
  update $x^{(k+1)} \leftarrow (1 - \gamma^{(k)}) x^{(k)} + \gamma^{(k)} s^{(k)}$
end for

This algorithm proceeds as follows. Starting with a feasible point $x^{(0)} \in D$, then in each step k, the algorithm computes the linear approximation of the objective function f evaluated in any point s that belongs the neighborhood of $x^{(k)}$. This approximation function is $f(x^{(k)}) + \nabla f(x^{(k)})^T (s - x^{(k)})$. Then, the algorithm finds the minimium of this function, which consists in solving the problem

$$s^{(k)} = \underset{s \in D}{\operatorname{argmin}} [f(x^{(k)}) + \nabla f(x^{(k)})^T (s - x^{(k)})] = \underset{s \in D}{\operatorname{argmin}} \nabla f(x^{(k)})^T s.$$

This problem is an easier problem than the original one ((II.9)), because its objective function is linear. Then, the algorithm moves in the direction of this minimizer as illustrated in Figure II.1([Jaggi 13]). Nevertheless, the algorithm does not takes directly the minimizer $s^{(k)}$, but also considers a part of the previous solution $x^{(k)}$, and this is to go slowly to convergence, and avoid big oscillations. More precisely, it takes a convex combination of the actual direction $s^{(k)}$ and the previous solution $x^{(k)}$. Due to the convexity of the space D, this convex combination belongs to D, since both $s^{(k)}$ and $x^{(k)}$ belong to D.

About the step size of the algorithm $\gamma^{(k)}$, it admits several variants. The simplest one is $\gamma^{(k)} = \frac{2}{k+2}$. It permits to take a fewer and fewer part of the minimizer as the algorithm proceeds. A more advanced one is the step size by line search:

$$\gamma^{(k)} = \underset{\alpha \in [0,1]}{\operatorname{argmin}} f((1 - \alpha) x^{(k)} + \alpha s).$$

This variant chooses at each iteration the best step size, in the sense that it the objective function at the iteration.

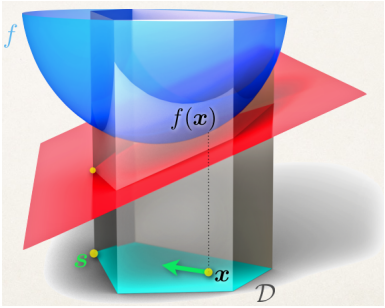


Figure II.1: Illustration of Frank-Wolfe: the red plan is the linear approximation of the function in blue at $f(x)$. s is the point that minimizes the red plan while satisfying the constraints represented by the space in turquoise. Figure taken from [Jaggi 13].

As it is shown in the previous description of the algorithm, the main advantage of the Frank-Wolfe Algorithm, as opposed to the projected gradient descent algorithm, is that there are no projections for the updated position on the convex set D at each step k, since the step is directly a minimum over the constraint set. Another interesting point on the computational side is that at each step, we only need to solve an optimization problem with a linear objective function.

In terms of convergence of Algorithm 1, it has been proved that for all k, we have

$$f(x^{(k)}) - f(x^*) \leq O(\frac{1}{k}).$$

II.4.3 A Frank-Wolfe based algorithm

Recall that our objective is to solve:

$$\min_{x \in X} g(x), \tag{II.10}$$

where

$$g(x) = \mu^T x + \sqrt{x^T \Sigma x}.$$

Note that Problem (II.10) is a constrained binary non-linear problem. In order to define the gradient needed in Algorithm 2, we use the fact that Σ symmetric positive definite.This means that $x^T \Sigma x$ is null only if x equals $0_{\mathbb{R}^m}$ that does not belong to X thanks to Assumption **(A3)**. So for all x in X, the gradient of g at x is computed as follows:

$$\nabla g(x) = \mu + \frac{\Sigma x}{\sqrt{x^T \Sigma x}}.$$

One should notice that we are not able to directly use the Frank-Wolfe algorithm directly to solve Problem II.10, since the constraint set X is discrete and so the problem is not convex. Instead, we propose a heuristic algorithm based on Frank-Wolfe. The choice to work on Frank-Wolfe Algorithm is not random, but it is natural to think of this algorithm, since the objective function g is convex, which fulfills

one assumption for using this algorithm. Next, this algorithm is usually used for optimization problems with the same form, but with continuous constraints. The closest example to our work in the state-of-the-art is the one in [Buchheim 18a], where they use Frank-Wolfe Algorithm for solving the relaxed robust knapsack problem under ellipsoidal uncertainty. The proposed heuristic approach we propose is denoted as DFW Algorithm referring to Discrete Frank-Wolfe, and it is described in Algorithm 2.

Algorithm 2 DFW: a Frank-Wolfe based algorithm to solve Problem (II.10)

1: $x^{(0)}$ a random feasible solution, $\varepsilon > 0$ close to zero, K a maximum number of iterations.
2: $k \leftarrow 1$
3: stop $\leftarrow$ false
4: **while** $k \leq K$ and $\neg$stop **do**
5: **if** $g(x^{(k-1)}) - g(x^{(k)}) < \varepsilon$: **then**
6: stop $\leftarrow$ true
7: **else**
8: $s^{(k)} \in \underset{y \in \text{Conv}(X)}{\text{argmin}} \nabla g(x^{(k)})^T y$, with $s^{(k)} \in X$
9: $\gamma^{(k)} \leftarrow \underset{\alpha \in [0,1]}{\text{argmin}}\, g(x^{(k)} + \alpha(s^{(k)} - x^{(k)}))$
10: $x^{(k+1)} \leftarrow x^{(k)} + \gamma^{(k)}(s^{(k)} - x^{(k)})$
11: **end if**
12: $k++$
13: **end while**
14: return $\underset{s \in \{s^{(1)}, \ldots, s^{(k-1)}\}}{\text{argmin}} g(s)$

Note k equals k_{end} at the last iteration of DFW algorithm.

The main idea of our approach is that we use the classic Frank-Wolfe algorithm on the convex hull $\text{Conv}(X)$, and we exploit the fact that $s^{(k)}$ belongs to X thanks to **(A2)**, which means that it is a feasible solution. When Algorithm 2 stops, $x^{(k)}$ is close to the optimal solution x^* of the relaxed problem in the sense that $g(x^{(k)}) - g(x^*) \leq O(\frac{1}{k})$.

Adding to this that $s^{(k)}$ (Algorithm 2, Line 8) is a minimizer of the linear approximation of g in the neighbourhood of $x^{(k)}$, and $x^{(k)}$ tends to minimize g in $\text{Conv}(X)$. This leads us to think that $\text{argmin}_{s \in \{s^{(1)}, \ldots, s^{(k_{\text{end}})}\}} g(s)$ is the best choice to minimize g in X. In other words, DFW Algorithm is a smart generator of good candidates for the optimization problem. We choose the line search step $\gamma^{(k)}$ (Algorithm 2, Line 9) because it guarantees that $g(x^{(k)})$ decreases at each iteration. The stopping criteria has been chosen as the convergence of the relaxed problem.

The following section shows numerical results that validate the heuristic proposed approach DFW.

II.5 Numerical results

This section is dedicated to illustrate the results of Algorithm 2. First, we describe the experimental setup. Then we discover the evolution of some interesting metrics along the iterations of the algorithm, and we finally compare the solutions and the performance between DFW and the CPLEX solver as a function of the size of the problem.

II.5.1 Experimental setup

To test the algorithm, we take the example of the shortest path problem that can be written in the form (II.1) and verifies the assumptions **(A1)**, **(A2)** and **(A3)**. We consider an undirected graph G with n nodes and m edges, and we would like to find a path between a source node s and a destination node d (s — d-path) with minimal cost, where the costs associated to the edges can be, e.g., the duration or the distance between the nodes. In this type of problems, we have:

$$X = \{x \in \{0,1\}^m; Ax = b\},$$

where

- A is the incidence matrix (of size $n \times m$),
- $b \in \mathbb{R}^m$ is given by $b_i = \mathbb{1}_{\{i=s\}} - \mathbb{1}_{\{i=d\}}$, $i = 1, \dots, m$, which means that it is a vector with the value 1 in the index of the source node s, -1 in the index of the destination node d, and 0 elsewhere.

We choose to take grid graphs (illustrated in Figure II.2) with L rows and L columns so that $n = L^2$ and $m = 4L(L-1)$. Note that the number of edges m is $2L(L-1)$, but it is doubled since it is an undirected graph, and thus every edge counts two edges, since there are two ways from each node to the other. We choose that the source and destination node be $s = (1,1)$ and $d = (L,L)$ respectively, as represented in Figure II.2 .

In the following numerical illustrations, we take $\mu = (\mu_1, \dots, \mu_m)$ where μ_i is chosen randomly in $[0, 100]$, $i = 1, \dots, m$. The random covariance matrix Σ is defined as the following: writing $\Sigma = \mathcal{P}^T \mathcal{D} \mathcal{P}$ where $\mathcal{P}$ is an orthogonal eigenvector matrix and $\mathcal{D}$ is the corresponding diagonal eigenvalue matrix, each of the eigenvalues λ_i, $i = 1, \dots, m$, is chosen as the square of a random number in $[0, \mu_i]$ and $\mathcal{P}$ is a random orthogonal matrix. This covariance matrix definition has been taken from [Ilyina 17].

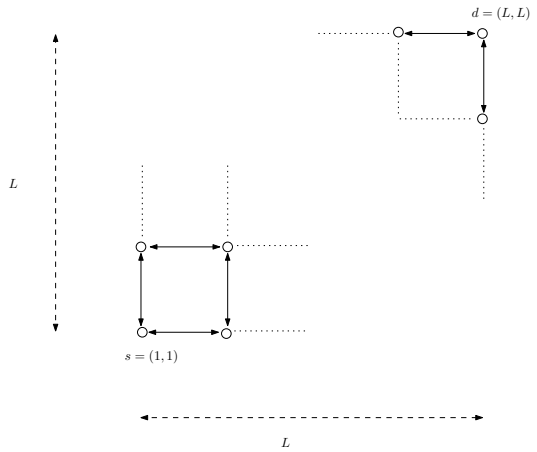


Figure II.2: Grid graph model.

For the implementation, we use the `Python` language, with the `Networkx` package for creating and manipulating graphs. To compute the solution $s^{(k)}$ (Line 8 of Algorithm 2), we used a Linear Programming (LP) minimizer with the LP modeler `PuLP`. Computations have been performed on an Intel® Core™ i5-8350U CPU @ 1.70GHz × 8.

In all the results of this paper, we set $\varepsilon = 10^{-6}$, $K = 1000$ and $\Omega = 1$. The random feasible solution $x^{(0)}$ in the case of the shortest path problem is a random path. To generate one, we generate a random cost vector c, and then we solve the shortest path problem (Problem (II.1)) using Dijkstra's Algorithm [Dijkstra 59] or the LP minimizer with the LP modeler `PuLP` to find a feasible solution $x^{(0)}$. The choice of Ω has been made for analogy with [Ilyina 17], and, with the range of the generated values of μ and Σ, it makes the problem hard enough to make the validation of DFW Algorithm possible.

II.5.2 Behavior of DFW algorithm

This section aims to observe the behavior of DFW Algorithm.

II.5.2.a Behavior for a grid graph size $L = 34$

To observe the behavior of DFW algorithm, we take the grid graph with $L = 34$. We denote, at each iteration k, the best solution so far as:

$$s_{opt}^{(k)} = \underset{s^{(l)} \in \{s^{(1)}, \ldots, s^{(k)}\}}{\operatorname{argmin}} g(s^{(l)}).$$

Note then that $s_{opt}^{(k_{\text{end}})}$ is the heuristic solution proposed by DFW. We show in Figure II.3 the evolution of $g(s^{(k)})$ and $g(s_{opt}^{(k)})$ in the 200 first iterations.

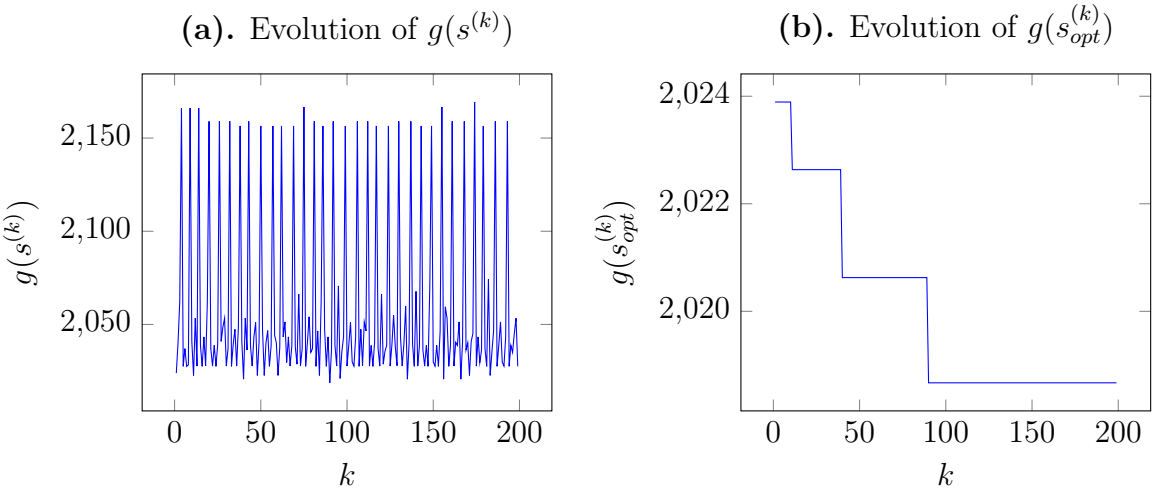


Figure II.3: The evolution of $g(s^{(k)})$ and $g(s_{opt}^{(k)})$ for $L = 34$ in the 200 first iterations.

L	k_1
5	16
6	43
7	2 & 41 & 111
8	267
9	229
10	7
11	38

Table II.1: The iteration k_1 of obtaining the optimal solution for different problem sizes.

In this example, the algorithm gives the same solution as CPLEX at iteration $k = 90$ (Figure II.3.b.). We see that, although $s^{(k)}$ alternates all along the iterations (Figure II.3.a.), it discovers new best obtained values as $g(x^{(k)})$ decreases and gets closer to the optimal solution $g(x^*)$.

II.5.2.b Number of iterations to get to convergence

Here, we show for different problem sizes L ($L \in \{5, 6, \ldots, 11\}$), the iteration k_1 at which DFW Algorithm gives the optimal solution. It is detailed in Table II.1. These results vary depending on the initial random path $x^{(0)}$. Even for a fixed problem size (e.g., $L = 7$), changing the initial solution three times gives respectively three different values of the iteration of obtaining the optimal solution, which are respectively k_1: 2, 41 and 111.

II.5.3 Performance of the DFW algorithm as a function of L

In order to evaluate the DFW Algorithm performance, we change the size of the graph, and we compare with the solutions provided by CPLEX. Experiments show that for small to medium graphs, DFW gives the same solution as CPLEX. Arguably, when L is large, methods based on branch-and-bound are no more efficient. In fact, in graphs corresponding to values of L larger than 40, due to the increase of memory consumption, the console only displays at the end the cost of the best integer solution found so far, rounded off to 4 decimal digits, without the corresponding integer vector. Hence we could only compare with this value and thus we could not use CPLEX to obtain a robust solution. For instance, in the case of $L = 40$, for CPLEX, the process stopped after two hours, without concluding if the best integer rounded to 2412.2594 corresponds to the optimal solution or not. Whereas, DFW Algorithm gives the solution 2412.2594001669304 at iteration 16 only, as shown in Figure II.4 (200 iterations takes half an hour). The comparison with the rounded digit of CPLEX makes us think that this is the optimal solution.

We tested graphs with L up to 46, ($n = 2116$ and $m = 4140$), and we have observed that the cost of the solution proposed by DFW is the same obtained by CPLEX. This result demonstrates that, even if we are not able to prove formally that DFW gives the optimal solution, it is a heuristic approach that is indeed efficient in cases of big graphs, where branch-and-bound methods are no more efficient.
Another interesting fact to mention is that, in more than 97% of the cases, $\varepsilon = 10^{-3}$ is more than enough to obtain the same solution given by CPLEX, and in all our experiments, we obtained it with less than 300 iterations.

II.5.4 Synthesis

To sum up the numerical results, the behavior of DFW algorithm is controlled, and it surpassed the BSOCP solver of CPLEX. These numerical findings show that the approach is promising. To understand the importance of the size of graphs taken in the study, one may consider the example of the city of Barcelona. It can be represented with a graph of 1020 nodes and 2522 edges. Another example is Berlin-Mitte-Center which can be represented with 398 nodes and 871 edges. This is to be compared with a grid graph with $40 \times 40 = 1600$ nodes and 3120 edges.

In this chapter, the heuristic approach DFW has been presented and a first validation has been led by comparison with the exact method. But as we saw in the numerical part, the exact method has the limitation of not being applicable in problems with large size, such as for the case of the robust shortest path problem

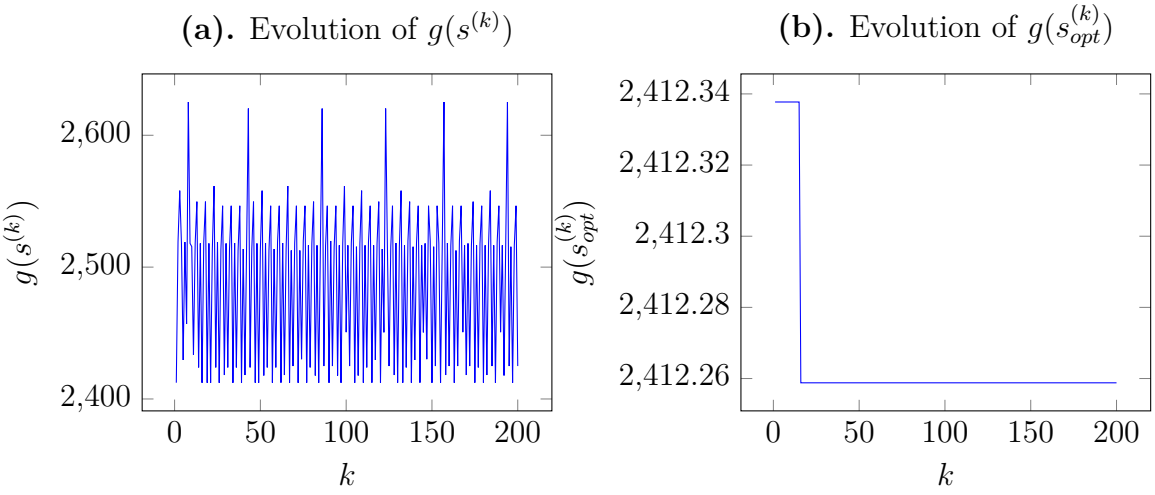


Figure II.4: The evolution of $g(s^{(k)})$ and $g(s_{opt}^{(k)})$ for $L = 40$ in the 200 first iterations.

with the grid graph of 40×40 nodes. This motivates the following chapter, where we are interested in finding another validation method than the comparison with exact methods.

Chapter III

Validation method for the heuristic solution applied on the robust shortest path problem

Recall from Chapter II that the first way to evaluate the quality of the solution given by the DFW Algorithm or any other approach that solves Problem (II.6) is to compare it with the solution given by the optimal solution of the BSOCP using an exact solver like CPLEX. Since this approach is no longer usable when considering large size problem and in order to be independent of exact solvers, the second contribution of this thesis is described in this chapter, which aims to develop a method to evaluate the quality of the heuristic approach. This method consists in computing a lower bound by a semi-definite relaxation of the robust problem. The gap between the solution proposed by the heuristic approach and the lower bound is a metric to evaluate the quality of the heuristic approach, since this gap contains the gap between the heuristic solution and the optimal one. The relaxed problem results from a bidualization that is done by a reformulation of the robust problem into a quadratic problem. In order to find this lower bound, a Semi-Definite Programming problem (SDP) needs to be solved. Unfortunately, the bidual problem is a big problem with much more constraints and more variables than the original problem. Thus, despite its polynomial nature, the resolution of this bidual problem is very time consuming and needs a huge memory space. Therefore, the sparsity of the matrices that define the problem is exploited to replace the classical solver by a sparse version of Pierra's decomposition through formalization in a product space algorithm. All this is numerically tested on the robust shortest path problem, showing that, with these results, a polynomial time evaluation of the quality of the solution of DFW heuristic is possible without having the memory storage issue of the bidual problem.

Notation

Throughout this chapter, we use the following matrix notation. Unless stated otherwise, all vectors belonging to $\mathbb{R}^l$ for some $l \in \mathbb{N}^*$ are column vectors. Furthermore, for some matrix M, $M[a:b, c:d]$ denotes, for all integers $a \leq b$ and $c \leq d$, the sub-block containing the entries in the rows a to b and columns c to d. $M[a, c:d]$ (resp. $M[a:b, c]$) is short for $M[a:a, c:d]$ (resp. $M[a:b, c:c]$). M^T is the transpose of M. Finally, $0_{(l,l)}$ is the block of dimension $l \times l$ with zeros everywhere and I_l is the identity block of dimension $l \times l$.

III.1 Evaluation of the quality of the approximate solution

This section starts with the general method of bidualization of any quadratic problem. Then, it describes the transformation of Problem (II.6) into a quadratic problem to bidualize. This gives the bidual problem that is an SDP problem. Then, in the next part, we show how to solve this SDP problem, first with an exact method, then with an adapted method to encounter the challenges of the SDP problem in hand.

III.1.1 Bidualization of a quadratic problem

Before giving a lower bound for Problem (II.6), we state a lower bound by bidualization for any quadratic problem. Then, we find a way to write Problem (II.6) as a quadratic problem following the general form.

A lower bound for quadratic programming problems is proposed in [Lemaréchal 99]. The lower bound proposed in this reference is the result of a bidualization. A first dualization of the quadratic problem is done applying the Lagrangian duality([Reeves 93]). This gives a dual problem that is a first SDP problem that gives a first lower bound. Moreover, Lemaréchal and Oustry studied the dualization of the dual problem using SDP duality, that is an extension of Lagrangian duality over SDP problems. They proved that doing this gives a bidual problem (also an SDP problem) that gives a second lower bound for the quadratic problem. This second lower bound is even better than the first one, in the sense that the second one is closer to the optimal solution of the quadratic problem than the first one. The authors also show that this bidualization procedure is nothing but the very known SDP relaxation, as we see in the following.

Consider the quadratic problem with N constraints:

$$\begin{aligned} \inf q_0(x), \quad & x \in \mathbb{R}^d \\ q_j(x) = 0, \quad & j = 1, \dots, N, \end{aligned} \tag{III.1}$$

where

$$q_j(x) = x^T Q_j x + b_j^T x + c_j, \quad j = 0, \dots N$$

are $N+1$ quadratic functions defined on $\mathbb{R}^d$, $d \in \mathbb{N}^*$ being the dimension of the problem, with the matrices Q_j lying in the set S^d of symmetric matrices of size $d \times d$, the values b_j in $\mathbb{R}^d$, and the values c_j in $\mathbb{R}$ $\forall j \in \{1, 2, \dots, N\}$; it is assumed that $c_0 = 0$.

Applying the Lagrangian duality on Problem (III.1) gives the dual problem:

$$\begin{aligned} &\sup r, \\ &u \in \mathbb{R}^N, r \in \mathbb{R}, \\ &\begin{bmatrix} c(u) - r & \frac{1}{2}b(u)^T \\ \frac{1}{2}b(u) & Q(u) \end{bmatrix} \succeq 0, \end{aligned} \tag{III.2}$$

where $Q(u) = Q_0 + \Sigma_{j=1}^N u_j Q_j$, $b(u) = b_0 + \Sigma_{j=1}^N u_j b_j$, and $c(u) = \Sigma_{j=1}^N u_j c_j$, and where the notation $M \succeq 0$ means that M is positive semi-definite, for any symmetric matrix M.

Then, applying SDP duality for the dual problem (III.2) reproduces the bidual problem of Problem (III.1) that is given by

$$\begin{aligned} \inf Q_0 \bullet X + b_0^T x, \quad & X \in S^d, x \in \mathbb{R}^d, \\ Q_j \bullet X + b_j^T x + c_j = 0, \quad & j = 1, \ldots, N, \\ & \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0, \end{aligned} \tag{III.3}$$

where the inner product between the matrices A and B of size $d \times d$ is defined by

$$A \bullet B = tr(A^T B). \tag{III.4}$$

This bidualization has another interpretation: it is also a direct convexification or SDP relaxation of Problem (III.1). Indeed, noticing that Problem (III.1) can also be written as

$$\begin{aligned} \inf Q_0 \bullet X + b_0^T x, \quad & X \in S^d, x \in \mathbb{R}^d, \\ Q_j \bullet X + b_j^T x + c_j = 0, \quad & j = 1, \ldots, N, \\ & X = xx^T, \end{aligned} \tag{III.5}$$

by setting $X = xx^T$, and writing a quadratic form $x^T Q x$ as $Q \bullet xx^T$, and then relaxing the nonconvex constraint $X = xx^T$ to $X \succeq xx^T$, that is convex with respect to (x, X). Then, we can see the previous bidualization as a convexification. Here we also use the equivalence(III.6) that can also be found in [Lemaréchal 99, lemma 3.9].

For any $(X, x) \in (S^d \times \mathbb{R}^d)$

$$X \succeq xx^T \iff \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0. \tag{III.6}$$

The inequality deduced from this bidualization is the following. If p^* is the optimal solution of the quadratic problem (III.1), and d^{**} is the optimal solution of its bidual problem (III.3), then the following inequality holds(see [Lemaréchal 99,

Proposition 4.5]:

$$d^{**} \leq p^*. \tag{III.7}$$

This inequality holds simply because, as already mentioned, Problem (III.3) is a relaxation for Problem (III.1). Note that d^{**} is a better bound than the solution d^* of the first dual problem (III.2) because $d^* \leq d^{**} \leq p^*$. The first inequality holds because of the weak quality property of the Lagrangian duality, with a change of signs. This is the main interest of bidualizing: we get a better lower bound than if we simply dualize our problem. An important note is that for combinatorial problems like the problems we consider, the problems are discrete, and thus non convex. Thus, the zero gap has no theoretical guaranties. It could happen by chance, but it is not a sure fact.

Hence, solving the SDP problem (III.3) enables to obtain a good lower bound for p^*. In general, this technique is used when we do not have an optimal solution for Problem (III.1), but instead we have a heuristic method for solving it and we want to validate this method. In this case, we solve Problem (III.3), that is easier, since it is a convex problem. Solving Problem (III.3) gives the lower bound d^{**}. The distance between the lower bound and the heuristic solution indicates how far this heuristic solution is from the optimal solution.

III.1.2 Using the bidualization to compute a lower bound

This section shows how to use the bidualization of general quadratic problems to compute a lower bound for our problem.

III.1.2.a Bidualization of the addressed problem

We now come to show how to use the bidualization, explained in Section III.1.1, to compute a lower bound for Problem (II.6). Recall that Problem (II.6) has another formulation, that is a BSOCP (Problem (II.8)). Rewriting Problem (II.8) more explicitly gives

$$\begin{aligned} \min \quad & \mu^T x + z \\ \text{s.t.} \quad & \sqrt{y^T y} \leq z \\ & y = (\Sigma^{\frac{1}{2}})^T x \\ & x \in X, y \in \mathbb{R}^m, z \in \mathbb{R}_+. \end{aligned} \tag{III.8}$$

First, the BSOCP formulation (III.8) of Problem (II.6) can be written as a Binary Quadratic Problem (BQP) since the variables y and z in Problem (III.8) are such that $\sqrt{y^T y} \geq 0$ and $z \geq 0$ for any $y \in \mathbb{R}^m$ and $z \in \mathbb{R}_+$. Thus, Problem (III.8) is

equivalent to

$$\begin{aligned} \min \quad & \mu^T x + z \\ \text{s.t.} \quad & y^T y \leq z^2 \\ & y = (\Sigma^{\frac{1}{2}})^T x \\ & x \in X, y \in \mathbb{R}^m, z \in \mathbb{R}_+. \end{aligned} \tag{III.9}$$

Before proceeding, it is important to mention that it has been proved here that the BSOCP problem can been written as a BQP problem. Nevertheless, it does not mean that the problem could have been easier to solve with this formulation than with the formulation in Section II.3. Indeed, in combinatorial optimization, it is known that a problem could be written in many forms. But this does not necessarily mean that it is easier to solve in a form than another.

Let us proceed with the formulation of Problem (III.9) as a problem in the form (III.1). This is done by writing all the constraints in the form of equalities. For this, we first can write

$$x \in X \iff Ax = b \text{ and } x \in \{0,1\}^m \iff Ax = b \text{ and } x_i(x_i - 1) = 0 \quad i = 1, \ldots, m.$$

Second, the inequality $y^T y \leq z$ can be transformed into an equality by considering additional variables c_1 and c_2 as follows:

$$y^T y \leq z^2 \iff \exists c_1 \in \mathbb{R} \; ; \quad y^T y - z^2 = -c_1^2 \iff \exists c_1 \in \mathbb{R} \; ; \quad y^T y - z^2 + c_1^2 = 0$$
$$z \geq 0 \iff \exists c_2 \in \mathbb{R} \; ; \quad z = c_2^2 \iff \exists c_2 \in \mathbb{R} \; ; \quad z - c_2^2 = 0.$$

Problem (III.9) is then equivalent to the following problem

$$\begin{aligned} \min \quad & \mu^T x + z \\ \text{s.t.} \quad & y^T y - z^2 + c_1^2 = 0 \\ & y = (\Sigma^{\frac{1}{2}})^T x \\ & Ax = b \\ & x_i(x_i - 1) = 0 \quad i = 1, \ldots, m \\ & z - c_2^2 = 0 \\ & x \in \mathbb{R}^m, y \in \mathbb{R}^m, z \in \mathbb{R}, c_1 \in \mathbb{R}, c_2 \in \mathbb{R}. \end{aligned} \tag{III.10}$$

We now need to write Problem (III.10) in a more compact way , i.e., in function of one variable, vector $u = [x, y, z, c_1, c_2] \in \mathbb{R}^{2m+3}$, and write each constraint individually. This makes Problem (III.10) equivalent to

$$
\begin{array}{ll}
\min & (\tilde{\mu} + \delta_{2m+1})^T u \\
\text{s.t.} & u^T(\mathbb{1}_y^T \mathbb{1}_y - \delta_{2m+1\ 2m+1} + \delta_{2m+2\ 2m+2})u = 0 \\
& (\mathbb{1}_y - (\widetilde{\Sigma^{\frac{1}{2}T}}))_i^T u = 0 \quad i = 1, \dots, m \\
& \tilde{A}_j^T u = b_j \quad j = 1, \dots, n \\
& u^T \delta_{ii} u - \delta_i^T u = 0 \quad i = 1, \dots, m \\
& -u^T \delta_{2m+3\ 2m+3} u + \delta_{2m+1}^T u = 0 \\
& u \in \mathbb{R}^{2m+3},
\end{array}
\tag{III.11}
$$

where the vectors and matrices that appear in Problem (III.11) are defined as follows

- the vector $\tilde{\mu}$ of size $2m+3$ is defined block-wise as $\tilde{\mu} = [\mu, 0, \dots, 0]^T$, so that $\tilde{\mu}^T u = \mu^T x$ if $u = [x, y, z, c_1, c_2]$,
- for any $k = 1, \dots, 2m+3$, $\delta_k \in \mathbb{R}^{2m+3}$ is such that $\delta_k(l) = 1$ if $k = l$, and 0 if else, so that $\delta_{2m+1}^T u = u_{2m+1} = z$, and $\delta_i^T u = x_i$ for $i = 1, \dots m$,
- $\mathbb{1}_y$ is an $m \times (2m+3)$ matrix such that $\mathbb{1}_y[m+1 : 2m; m+1 : 2m] = I_m$ and 0 elsewhere, so that $\mathbb{1}_y u = y$ and $u^T \mathbb{1}_y^T \mathbb{1}_y u = y^T y$,
- for any $i, j = 1, \dots, 2m+3$, $\delta_{i,j}$ is a $(2m+3) \times (2m+3)$ matrix, such that $\delta_{i,j}(k, l) = 1$ if $i = k$ and $j = l$, and 0 if else. So that $u^T \delta_{2m+1,2m+1} u = z^2$, $u^T \delta_{2m+2,2m+2} u = c_1^2$, $u^T \delta_{2m+3,2m+3} u = c_2^2$ and $u^T \delta_{ii} u = u_i^2$ for $i = 1, \dots, m$,
- $\widetilde{\Sigma^{\frac{1}{2}T}}$ is an $m \times (2m+3)$ matrix such that $\widetilde{\Sigma^{\frac{1}{2}T}}[1 : m; 1 : m] = \Sigma^{\frac{1}{2}T}$ and the other entries are zeros, so that $\widetilde{\Sigma^{\frac{1}{2}T}} u = \Sigma^{\frac{1}{2}T} x$,
- $\tilde{A}$ is an $n \times (2m+3)$ matrix such that $\tilde{A}[1 : n; 1 : m] = A$ and the other entries are zeros, so that $\tilde{A}u = Ax$.

Then, the bidual problem of Problem (III.11) is the following

$$
\begin{array}{ll}
\min & (\tilde{\mu} + \delta_{2m+1})^T u \\
\text{s.t.} & (\mathbb{1}_y^T \mathbb{1}_y - \delta_{2m+1\ 2m+1} + \delta_{2m+2\ 2m+2}) \bullet U = 0 \\
& (\mathbb{1}_y - (\widetilde{\Sigma^{\frac{1}{2}T}}))_i^T u = 0 \quad i = 1, \dots, m \\
& \tilde{A}_j^T u = b_j \quad j = 1, \dots, n \\
& \delta_{ii} \bullet U - \delta_i^T u = 0 \quad i = 1, \dots, m \\
& -\delta_{2m+3\ 2m+3} \bullet U + \delta_{2m+1}^T u = 0 \\
& \begin{bmatrix} 1 & u^T \\ u & U \end{bmatrix} \succeq 0, \quad U \in S^{2m+3}, \quad u \in \mathbb{R}^{2m+3}.
\end{array}
\tag{III.12}
$$

The last step consists in writing Problem (III.12) in a compact way by the change of variable

$$Z = \begin{bmatrix} 1 & u^T \\ u & U \end{bmatrix} \in S^{2m+4}.$$

This can be done using the following changes:

1. For any $v \in \mathbb{R}^{2m+3}$, we write $v^T u = V \bullet U$, where $V \in S^{2m+4}$ is defined by

$$V = \frac{1}{2} \left[\begin{array}{c|c} 0 & v^T \\ \hline v & 0 \end{array}\right] \in S^{2m+4}.$$

2. For any $W \in S^{2m+3}$, we write $W \bullet U = \mathbf{W} \bullet Z$, where $\mathbf{W} \in S^{2m+4}$ is defined by

$$\mathbf{W} = \left[\begin{array}{c|c} 0 & \dots 0 \\ \hline 0 & W \end{array}\right].$$

Due to this change of variable, the bidual problem of Problem (III.9) can be written as an SDP problem in the more compact way:

$$\begin{array}{ll} \min & M \bullet Z \\ \text{s.t.} & Z \in S^{2m+4} \\ & O_j \bullet Z = b_j, j = 1, \dots, n, \\ & C_i \bullet Z = 0, i = 1, \dots, m, \\ & Q \bullet Z = 0, \\ & D_i \bullet Z = 0, \quad i = 2, \dots, m+1 \\ & R \bullet Z = 0, \\ & Z \succeq 0, \end{array} \tag{III.13}$$

where $M \in S^{2m+4}$ is defined as follows

$$M = \frac{1}{2} \left[\begin{array}{c|c} 0 & \mu^T 0 \dots 0 1 0 0 \\ \hline \mu & \\ 0 & \\ \vdots & \\ 0 & \quad 0 \\ 1 & \\ 0 & \\ 0 & \end{array}\right],$$

that is $M[1, 2 : m+1] = \frac{1}{2}\mu^T$, $M[1, 2m+2] = \frac{1}{2}$, $M[2 : m+1, 1] = \frac{1}{2}\mu$, $M[2m+2, 1] = \frac{1}{2}$, and zero elsewhere. The matrix $O_j \in S^{2m+4}$ is defined for all $j = 1, \ldots, n$ by

$$O_j = \frac{1}{2}\left[\begin{array}{c|c} 0 & A_j^T 0 \ldots 0\,0\,0\,0 \\ \hline A_j & \\ 0 & \\ \vdots & \\ 0 & \mathbf{0} \\ 0 & \\ 0 & \\ 0 & \end{array}\right],$$

that is $O_j[1, 2 : m+1] = \frac{1}{2}A_j^T$, $O_j[2 : m+1, 1] = \frac{1}{2}A_j$, and zero elsewhere. The matrix $C_i \in S^{2m+4}$ is defined for all $i = 1, \ldots, m$ by

$$C_i = \frac{1}{2}\left[\begin{array}{c|c} 0 & -(\Sigma^{\frac{1}{2}T})_i^T 0 \ldots 0\,1\,0 \ldots 0 \\ \hline -(\Sigma^{\frac{1}{2}T})_i & \\ 0 & \\ \vdots & \\ 0 & \\ 1 & \mathbf{0} \\ 0 & \\ \vdots & \\ 0 & \end{array}\right],$$

that is $C_i[1, 2 : m+1] = -\frac{1}{2}(\Sigma^{\frac{1}{2}T})_i^T$, $C_i[1, m+1+i] = \frac{1}{2}$, $C_i[2 : m+1, 1] = -\frac{1}{2}(\Sigma^{\frac{1}{2}T})_i$, $C_i[m+1+i, 1] = \frac{1}{2}$, and zero elsewhere. We also define the matrix Q by

$$Q = \left[\begin{array}{c|c|c|ccc} 0 & 0 \ldots 0 & 0 \ldots 0 & 0 & 0 & 0 \\ \hline & 0_{(m,m)} & & & & \\ \hline & & I_m & & & \\ \hline \vdots & & & -1 & & \\ & & & & 1 & \\ 0 & & & & & 0 \end{array}\right],$$

that is $Q[m+2 : 2m+1, m+2 : 2m+1]$ is the identity matrix of dimension m, $Q[2m+2, 2m+2] = -1$, $Q[2m+3, 2m+3] = 1$ and zero elsewhere. Next, for the definition of the matrices D_i, for every $i = 2, m+1$, D_i is a $2m+4 \times 2m+4$ matrix such that $D_i[i, i] = 1$, $D_i[i, 1] = -\frac{1}{2}$ and $D_i[1, i] = -\frac{1}{2}$. Finally, R is a $2m+4 \times 2m+4$ matrix such that $R[1, 2m+2] = \frac{1}{2}$, $R[2m+2, 1] = \frac{1}{2}$ and $R[2m+4, 2m+4] = -1$.

III.1.2.b The biduality gap

Now that we have stated the bidual problem (III.13) of Problem (III.9), the lower bound inequality (III.7) reads here

$$\text{val}((\text{III.13})) \leq \text{val}((\text{III.9})),$$

where val$((P))$ denotes the optimal value for a given problem (P). Due to the equivalence between Problem (II.6) and Problem (III.9), we know that val((III.9)) equals $g(x^*)$. This gives us an additional inequality:

$$\text{val}((\text{III.13})) \leq \text{val}((\text{III.9})) \ = g(x^*) \leq g(\hat{x}).$$

Or, written differently,

$$d^{**} \leq p^* = g(x^*) \leq g(\hat{x}). \tag{III.14}$$

Thus, d^{**} is a lower bound that allows to evaluate the quality of any heuristic solution, and thus the one obtained by DFW Algorithm. Hence, the biduality gap BG is defined as

$$BG = g(\hat{x}) - d^{**}. \tag{III.15}$$

A corresponding relative gap RBG is defined as

$$RBG = \frac{g(\hat{x}) - d^{**}}{d^{**}}. \tag{III.16}$$

Concerning this relative gap, the division could also have been possible by $g(\hat{x})$, since it also gives a relative gap, but we chose to divide by d^{**}, since usually, this is the adopted division for the relative gap in prediction algorithms: $\frac{\text{prediction - real value}}{\text{real value}}$.

Furthermore, and in order to analyze more precisely the obtained gap and to compare with other work, we are interested in another metric named *performance ratio* used by Karloff in [Karloff 99] for the *Max-Cut* problem. The *performance ratio* ρ is defined as follows:

$$\rho = \frac{d^{**}}{g(\hat{x})}. \tag{III.17}$$

We know that $d^{**} \leq g(\hat{x})$. Thus, $\frac{d^{**}}{g(\hat{x})} \leq 1$. So, the bigger ρ is, the closest it is to 1, and the better it is.

More explicitly, the validation process is the following: first solve the robust shortest path problem using the heuristic approach DFW and find a heuristic solution $\hat{x}$. Then evaluate the quality of this solution using Inequality (III.14) by proceeding as follows. We compute d^{**}, and if the gap between d^{**} and $g(\hat{x})$ is small, the gap between $g(x^*)$ and $g(\hat{x})$ is small too, since $g(\hat{x}) - d^{**} \geq g(\hat{x}) - g(x^*) \geq 0$. Then, a comparison between the performance ratio with the ones obtained in other work is done to go further in the evaluation process.

The only missing step now is to compute d^{**}. In the next section, we show how to solve Problem (III.13) to compute d^{**}.

III.1.3 Solving the SDP problem

The above sections aim at showing that a lower bound for the robust shortest path problem is the solution of an SDP problem that has to be solved. SDP is a particular class of convex optimization problems which appears in various engineering motivated problems, including the most efficient relaxations of some NP-hard problems such as often encountered in combinatorial optimization or mixed integer programming [Anjos 11]. SDP can be written as minimization over symmetric (resp. Hermitian) positive semi-definite matrix variables, with linear cost function and affine constraints, i.e., problems of the form:

$$\min_{Z \succeq 0} \left(\langle A, Z \rangle : \langle B_j, Z \rangle = b_j \text{ for } j = 1, \ldots, m \right), \tag{III.18}$$

where $A, B_1, \ldots, B_m$ are given matrices. Compact SDPs can be solved in polynomial time. SDP was extensively studied over the last three decades since its early use which can be traced back to [Scobey 78] and [Fletcher 81]. In particular, Linear Matrix Inequalities (LMI) and their numerous applications in control theory, system identification and signal processing have been a central drive for the use of SDP in the 90's as reflected in the book [Boyd 94]. One of the most influential paper for that era, is the one of Goemans and Williamson [Goemans 95] in which SDP was shown to provide a 0.87 approximation to the *Max-Cut* problem, a famous clustering problem on graphs. Other SDP schemes for approximating hard combinatorial problems have subsequently been devised for the graph coloring problem [Karger 98], for satisfiability problem [Goemans 95, Goemans 94]. These results were later surveyed in [Lemaréchal 99, Goemans 97] and [Wolkowicz 99]. Numerical methods for solving SDP's are manifold and various schemes have been devised for specific structures of the constraints. One of these families of methods is the class of interior point methods [Nesterov 94]. Such methods are known to be of the most accurate type, but suffer from being not scalable in practice. Another family of methods is based around the alternating direction method of multipliers (ADMM) technique [Boyd 11]. ADMM approaches are usually faster as they can be implemented in a distributed architecture. As such, they often appear to be faster and more scalable than interior point methods at the price of a worse accuracy. Other methods can also be put to work as the method of Pierra [Pierra 84] upon which we further elaborate in the present chapter.

As mentioned just above, interior point methods can be used to solve SDP problems, which gives a first way to solve the SDP problem (III.13): an option is to implement this resolution using the CVXPY Python package [Diamond 16]

which is a Python-embedded modeling language for convex optimization problems. CVXPY converts the convex problems into a standard form known as conic form, a generalization of a linear program. The conversion is done using graph implementations of convex functions. The resulting cone program is equivalent to the original problem, so by solving it we obtain a solution of the original problem. In particular, it solves the semi-definite programs using interior point methods. It is rather simple to use CVXPY to solve our SDP problem (III.13): define the function to be minimized, the constraints of the problem, and then launch the solver. However this simplicity has a price: the problem definition requires the storage of the matrices that describe the problem. More precisely, we need to store $n+2m+4$ matrices of dimension $2m+4 \times 2m+4$: one matrix to define the objective function, and $n+2m+3$ matrices for the constraints. This is a significant issue because of the storage necessity, especially in large size problems. To illustrate how big the storage grows with respect to the problem size, take a medium grid graph with 10×10 nodes ($n = 100$, $m = 360$). This problem size requires the storage of 824 matrices of dimension 724×724 (this takes 3.45 Gigabytes in double precision). A relatively big grid graph with 40×40 nodes ($n = 1600$, $m = 6240$) needs the storage of 14084 matrices of dimension 12484×12484 (this takes 17.5 terabytes in double precision). But since most of the matrices we have are sparse, we adopt another efficient approach where we are able to make sparse computations that allow us to avoid this main drawback considering the storage of the matrices. Before tackling this memory storage issue, we first describe the practical algorithm that we have implemented in order to find this d^{**}.

III.1.3.a Pierra's Decomposition through formalization in a product space

The goal of this section is to describe the algorithm of Pierra [Pierra 84] for optimization problems over an intersection of convex sets. But it is worth mentioning that the work of Pierra aimed first to find a point in an intersection of convex sets. This is stated and described in the following paragraph.

Find a point in an intersection of convex sets. Consider the following problem

$$\begin{aligned} &\text{Find } x \in \mathcal{H} \\ \text{s.t.} \quad &x \in \cap_{j=1}^{J} \mathcal{S}_j. \end{aligned} \tag{III.19}$$

The main idea of the work of Pierra is the formalization of the constraint set $\cap_{j=1}^{J} \mathcal{S}_j$ introducing the set $\mathbf{H} = \mathcal{H}^J$. Let us first define the sets $\mathbf{H}$, $\mathbf{S}$ and $\mathbf{D}$. First, the product set $\mathbf{H} = \mathcal{H}^J$ is such that

$$\mathbf{x} \in \mathbf{H} \iff \mathbf{x} = (x_1, \dots, x_J) \text{ with } x_1, \dots \text{ and } x_J \in \mathcal{H}.$$

Next, the subset $\mathbf{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_J \subset \mathbf{H}$ is such that

$$\mathbf{x} \in \mathbf{S} \iff \mathbf{x} = (x_1, \ldots, x_J) \text{ with } x_1 \in \mathcal{S}_1, \ldots \text{ and } x_J \in \mathcal{S}_J.$$

Now denote the diagonal convex $\mathbf{D} \subset \mathbf{H}$ such that

$$\mathbf{x} \in \mathbf{D} \iff \mathbf{x} = (x, \ldots, x) \text{ with } x \in \mathcal{H}.$$

Then, we have

Proposition 1.

$$x \in \cap_{j=1}^{J} \mathcal{S}_j \implies \boldsymbol{x} = (x, \ldots, x) \in \boldsymbol{S} \cap \boldsymbol{D}. \tag{III.20}$$

Proof. First, it is obvious that for any $x \in \mathcal{H}$, $\mathbf{x} = (x, \ldots, x) \in D$, by definition. Next, if $x \in \cap_{j=1}^{J} \mathcal{S}_j$, then $x \in \mathcal{S}_1$, ..., and $x \in \mathcal{S}_J$. Then $(x, \ldots, x) \in \mathbf{S}$. Thus, $\mathbf{x} = (x, \ldots, x) \in \mathbf{S} \cap \mathbf{D}$. □

Property (III.20) enables us to formalize Problem (III.19) in another space, which is the product space. Indeed, we have the equivalence between the two following problems:

$$\begin{aligned} &\text{In } \mathcal{H}: \\ &\qquad \text{Find } x \in \mathcal{H} \qquad\qquad (\text{III.21}) \\ &\text{s.t.} \quad x \in \cap_{j=1}^{J} \mathcal{S}_j. \\ &\text{In } \mathbf{H}: \\ &\qquad \text{Find } \mathbf{x} \in \mathbf{H} \qquad\qquad (\text{III.22}) \\ &\text{s.t.} \quad \mathbf{x} \in \mathbf{S} \cap \mathbf{D}. \end{aligned}$$

To find a solution of Problem (III.21) from a solution of Problem (III.22), we use the fact that a solution of Problem (III.22) belongs to $\mathbf{D}$, then it has the form $\mathbf{x} = (x, \ldots, x)$. Therefore, it is obvious that x is a solution of Problem (III.21).

This formalization reduces the problem, since in the product space $\mathbf{H}$, there are only two sets, instead of J sets in $\mathcal{H}$. In addition, the orthogonal projections over these two sets $\mathbf{S}$ and $\mathbf{D}$ is interesting, because of the following projection formulas

$$\textbf{(i)} Proj_{\mathbf{S}}(x, \ldots, x) = (Proj_{\mathcal{S}_1}(x), \ldots, Proj_{\mathcal{S}_J}(x)). \tag{III.23}$$

$$\textbf{(ii)} Proj_{\mathbf{D}}(x_1, \ldots, x_J) = (1/J\Sigma_{j=1}^{J} x_j, \ldots, 1/J\Sigma_{j=1}^{J} x_j). \tag{III.24}$$

$$\textbf{(iii)} Proj_{\mathbf{D}}(Proj_{\mathbf{S}}(x, \ldots, x)) = (1/J\Sigma_{j=1}^{J} Proj_{\mathcal{S}_j}(x), \ldots, 1/J\Sigma_{j=1}^{J} Proj_{\mathcal{S}_j}(x)). \tag{III.25}$$

The algorithm of Pierra described in Algorithm 3 is based on these successive projections: Lines 5 and 6 of Algorithm 3 are the successive projections over $\mathbf{S}$ and $\mathbf{D}$ stated in Formula (III.23) and then Formula (III.24). Most importantly,

it has been proved in [Pierra 84, Theorem 1.1] that the successive projections for Problem (III.21) give the method of successive projections for Problem (III.22) as it is described in Algorithm 3. To solve Problem (III.19), Pierra's Algorithm can be described in three steps:

(i) The first step (Line 5 of Algorithm 3) comes from the projection on **S**.

(ii) the second step comes from the projection over the diagonal convex **D**, represented in Line 6 of Algorithm 3.

(iii) Finally, the third step (Line 10 of Algorithm 3) is the extrapolation step. This gives the nomination of the method of Pierra, which is the *Extrapolated Parallel Projection Method* (EPPM). This step is a geometric way to make the successive projections faster, by taking a point b^{p+1} that belongs to $\{x^p+\theta(b'^{p+1}-x^p), \theta \geq 0\}$ and that has another property issued from the space **H** of which the explanation is not given here. Next, in simple words, the operation in Line 11 is used to center the iterate x^p from time to time, every k iterations: in [Pierra 84, Section 4], it is explained that without the centring technique, the convergence seems to become ineffective, and on the other side, centring at each iteration can lead to an ineffective extrapolation.

This algorithm converges, and the proof can be found in [Pierra 84, Theorem 1.2].

Algorithm 3 Pierra's algorithm to solve Problem (III.19)

1: $x^0 \in \mathcal{H}$ random, $k \in \mathbb{N}$, $\lambda \in]0,1]$, P the maximum number of iterations.
2: $p \leftarrow 0$
3: stop $\leftarrow$ false
4: **while** $p \leq P$ and $\neg$stop **do**
5: $\quad v_j^{p+1} \leftarrow Proj_{S_j}(x^p), j = 1, \ldots, J$
6: $\quad b'^{p+1} \leftarrow \frac{1}{J}\Sigma_{j=1}^{J} v_j^{p+1}$
7: $\quad$ **if** $b'^{p+1} = x^p$ **then**
8: $\quad\quad$ stop $\leftarrow$ true
9: $\quad$ **else**
10: $\quad\quad b^{p+1} \leftarrow x^p + \beta^{p+1}(b'^{p+1} - x^p)$ with $\beta^{p+1} \leftarrow \frac{\Sigma_{j=1}^{J}\|v_j^{p+1}-x^p\|^2}{J\|b'^{p+1}-x^p\|^2}$
11: $\quad\quad x^{p+1} \leftarrow \begin{cases} x^p + \lambda(b^{p+1} - x^p), & \text{if } p+1 \equiv k(\text{mod } k). \\ b^{p+1} & \text{otherwise.} \end{cases}$
12: $\quad\quad p++$
13: $\quad$ **end if**
14: **end while**
15: **return** x^p

Minimize a function over an intersection of convex sets. Here we describe the algorithm of Pierra [Pierra 84] for minimization problems over an intersection of convex sets.

For this, we consider a general minimization problem in a finite dimensional Hilbert space $\mathcal{H}$ equipped with a norm $\|.\|_2$. Suppose we want to solve

$$\begin{aligned} \min_{x\in\mathcal{H}} \quad & f(x) \\ \text{s.t.} \quad & x \in \cap_{j=1}^{J} \mathcal{S}_j, \end{aligned} \tag{III.26}$$

where f is a differentiable function, and $\mathcal{S}_1, \ldots, \mathcal{S}_J$ are convex subsets of $\mathcal{H}$. Thus in this part, in addition to finding a point in the intersection of the convex sets, there is an objective function f to minimize. Thus the main ideas of the algorithm 3 remain the same, with a difference in Line 5 that considers the minimization part. Pierra's Algorithm to solve Problem (III.26) is described in Algorithm 4. It is important to note that, unfortunately, in the reference that we followed([Pierra 84]), Algorithm 4 is not written in a straightforward way, but its full form could be deduced from [Pierra 84, Algorithm 3.2], with the explicit formula of v_j^{p+1} that is written at the end of [Pierra 84, Theorem 3.2].

Algorithm 4 Pierra's algorithm to solve Problem (III.26)

1: $x^0 \in \mathcal{H}$ random, $k \in \mathbb{N}$, $\lambda \in]0,1]$, ϵ small, P the maximum number of iterations.

2: $p \leftarrow 0$
3: stop $\leftarrow$ false
4: **while** $p \leq P$ and $\neg$stop **do**
5: $\quad v_j^{p+1} \leftarrow Prox_{I_{\mathcal{S}_j} + \frac{1}{2J}\epsilon f}(x^p), j = 1, \ldots, J$
6: $\quad b'^{p+1} \leftarrow \frac{1}{J}\Sigma_{j=1}^{J} v_j^{p+1}$
7: $\quad$ **if** $b'^{p+1} = x^p$ **then**
8: $\quad\quad$ stop $\leftarrow$ true
9: $\quad$ **else**
10: $\quad\quad b^{p+1} \leftarrow x^p + \beta^{p+1}(b'^{p+1} - x^p)$ with $\beta^{p+1} \leftarrow \frac{\Sigma_{j=1}^{J}\|v_j^{p+1} - x^p\|^2}{J\|b'^{p+1} - x^p\|^2}$
11: $\quad\quad x^{p+1} \leftarrow \begin{cases} x^p + \lambda(b^{p+1} - x^p), & \text{if } p+1 \equiv k (\text{mod } k). \\ b^{p+1} & \text{otherwise.} \end{cases}$
12: $\quad\quad p++$
13: $\quad$ **end if**
14: **end while**
15: **return** x^p

In this algorithm, the proximal function associated to a function $h : \mathcal{H} \longrightarrow \mathbb{R}$ is given by

$$Prox_h(y) = \underset{x\in\mathcal{H}}{\operatorname{argmin}} \left[h(x) + \frac{1}{2}\|x - y\|_2^2 \right]. \tag{III.27}$$

Next, $I_{\mathcal{S}_j}(x)$ is the indicator function for the set $\mathcal{S}_j$, it equals 0 if $x \in \mathcal{S}$ and $+\infty$ otherwise. Finally, $\epsilon > 0$ is a tuning parameter for the minimization step which value is small (e.g., $\epsilon = 10^{-4}$).

Just like the previous paragraph that concerns finding a point in a intersection of convex sets, the main idea of the algorithm comes from the formalization of the constraint set $\cap_{j=1}^{J}\mathcal{S}_j$ introducing the set $\mathbf{H} = \mathcal{H}^J$, and then we reformulate Problem (III.26) in $\mathbf{H}$ as a minimization problem over $\mathbf{S} \cap \mathbf{D}$. The three steps are the same as in the previous paragraph with a difference in the first one: here, the first step described in Line 5 of Algorithm 4 comes from the projection on $\mathbf{S}$, with a part of minimization of the objective function. The role of the proximal function can be explained intuitively as follows: for every constraint space $\mathcal{S}_j$, it both minimizes the function f and stays close to x^p, and since x^p partially results from a point that belongs to all the constraint spaces, x^p converges to the optimal solution. The second and third step are the same as the ones described in the previous paragraph. It has been proved in [Pierra 84, Theorem 3.3] that this algorithm converges. All the theoretical background of Pierra's algorithm can be found in [Pierra 84].

III.1.3.b Pierra's algorithm adapted to solve our SDP problem

We now come to apply Algorithm 4 to solve Problem (III.13). In this case, the corresponding Hilbert space is set as $\mathcal{H} = S^{2m+4}$, with the norm $\|.\|_F$ that is associated to the inner product $\bullet$ defined in Section III.1.1 by Definition (III.4), such that $\|A\|_F^2 = tr(A^T A)$. The function to minimize in Problem (III.26) is given by $f : Z \in S^{2m+4} \mapsto f(Z) = M \bullet Z$, and the integer J equals $n + 2m + 3$. The convex sets $\mathcal{S}_1, \ldots, \mathcal{S}_J$ are defined as follows:

$$\begin{array}{rcll} \mathcal{S}_j & = & \{Z \in S^{2m+4};\ \mathbf{A}_j \bullet Z = \mathbf{b}_j\}, & j = 1, \ldots, n+2m+2, \\ \mathcal{S}_J & = & \{Z \in S^{2m+4};\ Z \succeq 0\}, & \end{array} \tag{III.28}$$

where $\mathbf{A}_j$, $\mathbf{b}_j$, $j = 1, \ldots, n + 2m + 2$ are respectively matrices and scalars defined by

$$\mathbf{A}_j = \begin{cases} O_j, & j = 1, \ldots, n, \\ C_{j-n}, & j = n+1, \ldots, n+m, \\ Q, & j = n+m+1, \\ D_{j-(n+m+1)}, & j = n+m+2, \ldots, n+2m+1, \\ R, & j = n+2m+2, \end{cases} \qquad \mathbf{b}_j = \begin{cases} b_j, & j = 1, \ldots, n, \\ 0, & j = n, \ldots, n+2m+2. \end{cases} \tag{III.29}$$

In our case, the proximal function associated to $I_{S_j} + \frac{1}{2J}\epsilon f$ on Line 5 of Algorithm 4 is computed using Definition (III.27) in the following way:

$$\begin{aligned} Prox_{I_{\mathcal{S}_j}+\frac{1}{2J}\epsilon f}(x^p) &= \underset{Z\in\mathcal{S}_j}{\operatorname{argmin}} \left[\frac{1}{2J}\epsilon M \bullet Z + \frac{1}{2}\|Z - x^p\|_F^2\right] \\ &= \underset{Z\in\mathcal{S}_j}{\operatorname{argmin}} \left[\frac{1}{2J}\epsilon M \bullet Z + \frac{1}{2}\|Z\|_F^2 - Z \bullet x^p + \frac{1}{2}\|x^p\|_F^2\right] \\ &= \underset{Z\in\mathcal{S}_j}{\operatorname{argmin}} \left[\frac{1}{2}\|Z\|_F^2 - Z \bullet (x^p - \frac{1}{2J}\epsilon M) + \frac{1}{2}\|x^p\|_F^2\right] \\ &= \underset{Z\in\mathcal{S}_j}{\operatorname{argmin}} \left[\frac{1}{2}\|Z - (x^p - \frac{1}{2J}\epsilon M)\|_F^2 = Proj_{\mathcal{S}_j}(x^p - \frac{1}{2J}\epsilon M)\right], \end{aligned} \tag{III.30}$$

where $Proj_{\mathcal{S}_j}$ is the projection on the set $\mathcal{S}_j$. Thus, one sees from (III.30) that there remains to compute the projections over the constraint spaces defined by Definitions (III.28). Those spaces are of two kinds. First, for any constraint in the form $C = \{Z \in S^{2m+4}; \mathbf{A} \bullet Z = \mathbf{b}\}$, we have the following explicit projection formula:

Proposition 2.

$$Proj_C(Z) = Z + \left(\frac{\boldsymbol{b} - \boldsymbol{A} \bullet Z}{\|\boldsymbol{A}\|_F^2}\right)\boldsymbol{A}.$$

Proof. This explicit projection formula is obtained using the following proof elements. First we have

$$Proj_C(Z) = \min_{Y\in C} \|Y - Z\|_F^2 = \min_{\substack{Y\in S^{2m+4} \\ \mathbf{A}\bullet Y - \mathbf{b} = 0}} \|Y - Z\|_F^2.$$

If we note $f(Y) = \|Y - Z\|_F^2$ and $g(Y) = \mathbf{A} \bullet Y - \mathbf{b}$, then the optimality conditions imply that this minima is obtained if and only if there exists a Lagrange multiplier [Nocedal 06] $\lambda \in \mathbb{R}$ such that $\nabla f - \lambda \nabla g = 0$, then there exists $\lambda \in \mathbb{R}$ such that $Y - Z = \lambda A$, which means that $Y = Z + \lambda A$. Next, to find λ, we use the fact that $Y \in C$, then

$$\begin{aligned} & A \bullet Y = b \\ \Longrightarrow\ & A \bullet (Z + \lambda A) = b \\ \Longrightarrow\ & A \bullet Z + \lambda A \bullet A = b \\ \Longrightarrow\ & \lambda = \frac{b - A \bullet Z}{\|A\|_F^2}. \end{aligned}$$

Then

$$Proj_C(Z) = Z + \left(\frac{\mathbf{b} - \mathbf{A} \bullet Z}{\|\mathbf{A}\|_F^2}\right)\mathbf{A}.$$

□

Second, concerning the projection on the constraint space $\mathcal{S}_J = \{Z \in S^{2m+4};\ Z \succeq 0\}$, we have

$$Proj_{\mathcal{S}_J}(Z) = U \max\{\Lambda, 0\} U^T,$$

where $Z = U\Lambda U^T$ is the eigenvector decomposition of the matrix Z (see [Anjos 11, section 20.1.1]).

In view of all these considerations, Pierra's algorithm applied on Problem (III.13) is described in Algorithm 5.

Algorithm 5 Pierra's algorithm to solve the SDP problem (III.13)

$Z^1 \in S^{2m+4}$ random, $k \in \mathbb{N}$, $\lambda \in]0,1]$, ϵ small, α small, P the maximum number of iterations.
$p \leftarrow 1$
stop $\leftarrow$ false
while $p \leq P$ and $\neg$stop **do**
  $Y^p \leftarrow Z^p - \frac{\epsilon}{2(n+2m+3)} M$
  for $j = 1$ to n **do**
    $Z_j^{p+1} \leftarrow Y^p + (\frac{b_j - O_j \bullet Y^p}{\|O_j\|^2}) O_j$
  end for
  for $i = 1$ to m **do**
    $Z_{n+i}^{p+1} \leftarrow Y^p + (\frac{-C_i \bullet Y^p}{\|C_i\|^2}) C_i$
  end for
  $Z_{n+m+1}^{p+1} \leftarrow Y^p + (\frac{0 - Q \bullet Y^p}{\|Q\|^2}) Q$
  for $i = 2$ to $m+1$ **do**
    $Z_{n+m+i}^{p+1} \leftarrow Y^p + (\frac{0 - D_i \bullet Y^p}{\|D_i\|^2}) D_i$
  end for
  $Z_{n+2m+2}^{p+1} \leftarrow Y^p + (\frac{0 - R \bullet Y^p}{\|R\|^2}) R$
  $Z_{n+2m+3}^{p+1} \leftarrow U^p \max\{\Gamma^{(p)}, 0\} U^{pT}$, where U^P (resp. Γ^p) are the eigenvectors (resp. the eigenvalues) of Y^p
  $B'^{p+1} \leftarrow \frac{1}{n+2m+3} \Sigma_{i=1}^{n+2m+3} Z_i^{p+1}$
  if $\|B'^{p+1} - Z^p\|^2 < \alpha$ **then**
    stop $\leftarrow$ true
  else
    $B^{p+1} \leftarrow \beta^{p+1} B'^{p+1} + (1 - \beta^{p+1}) Z^p$ with $\beta^{p+1} \leftarrow \frac{\Sigma_{i=1}^{n+2m+3} \|Z_i^{p+1} - Z^p\|^2}{(n+2m+3)\|B'^{p+1} - Z^p\|^2}$
    $Z^{p+1} \leftarrow \begin{cases} Z^p + \lambda(B^{p+1} - Z^p), & \text{if } p+1 \equiv k (\text{mod } k). \\ B^{p+1} & \text{otherwise.} \end{cases}$
    $p++$
  end if
end while
return Z^p

In order to solve Problem (III.13) using Algorithm 5, we need to store the matrices M, O_j $j = 1, \ldots, n$, C_i, $i = 1, \ldots, m$, Q, D_i, $i = 2, \ldots, m+1$ and R, that is in total $n + 2m + 4$ matrices of dimension $2m + 4 \times 2m + 4$. Nevertheless, there is a way to avoid storing these matrices, since in Algorithm 5, we do not need the whole matrices, but rather the result of operations that mostly include dot products of sparse matrices. So, if we compute the terms needed for Lines 5, 7, 10, 12, 14 and 16 of Algorithm 5 depending on A, b, μ, and Σ, then there is no need to store the matrices M, O_j $j = 1, \ldots, n$, C_i, $i = 1, \ldots, m$, Q, D_i, $i = 2, \ldots, m+1$ and R. All these calculations are detailed in the next section (III.1.3.c). This aspect is one of the contributions of this chapter.

III.1.3.c Sparse computations

The aim of this section is to detail the computations needed in Algorithm 5, and the replacements done to avoid the storage of the matrices M, O_j $j = 1, \ldots, n$, C_i, $i = 1, \ldots, m$, Q, D_i, $i = 2, \ldots, m+1$ and R. Recall that doing this enables us to express all the formulas depending only on A, b, μ, and Σ, and thus to avoid the storage of $n + 2m + 4$ matrices of dimension $2m + 4 \times 2m + 4$.

The operation in Line 5

1: $Y^p = Z^p - \frac{\epsilon}{2(n+2m+3)} M$

can be replaced by

1: $Y^p = Z^p$
2: ${Y^p}_{[1,2\to m+1]} = {Y^p}_{[1,2\to m+1]} - \frac{\epsilon}{4(n+2m+3)} \mu^T$
3: ${Y^p}_{[2\to m+1,1]} = {Y^p}_{[2\to m+1,1]} - \frac{\epsilon}{4(n+2m+3)} \mu$
4: ${Y^p}_{[1,2m+2]} = {Y^p}_{[1,2m+2]} - \frac{\epsilon}{4(n+2m+3)}$
5: ${Y^p}_{[2m+2,1]} = {Y^p}_{[2m+2,1]} - \frac{\epsilon}{4(n+2m+3)}$

Here, recall the form of the matrix M: $M[1, 2 : m + 1] = \frac{1}{2}\mu^T$, $M[1, 2m + 2] = \frac{1}{2}$, $M[2 : m+1, 1] = \frac{1}{2}\mu$, $M[2m+2, 1] = \frac{1}{2}$, and zero elsewhere. Then, in the operation $Y^p = Z^p - \frac{\epsilon}{2(n+2m+3)} M$, we initialize Y^p as Z^p, and we change only the elements of Y^p where M does not equal zero. The most important aspect for us is that storing the matrix M is useless, since we only need the vector μ. The same reasoning follows in the other operations.

The operation in Line 7

1: $Z_j^{p+1} = Y^p + (\frac{b_j - O_j \bullet Y^p}{\|O_j\|^2}) O_j$

can be replaced by

1: $Z_j^{p+1} = Y^p$
2: $Z_j^{p+1}[1, 2 : m + 1] = Z_j^{p+1}[1, 2 : m + 1] + \frac{a_j}{2} A_{j*}$

3: $Z_j^{p+1}[2:m+1,1] = Z_j^{p+1}[2:m+1,1] + \frac{a_j}{2}A_{j*}$,

with A_{j*} is the vector containing the j-th lign of A, $a_j = \frac{b_j - O_j \bullet Y^p}{\|O_j\|^2} = \frac{2b_j - \Sigma_{i=1}^m A_{ji}(Y^p_{i+1\ 1} + Y^p_{1\ i+1})}{\Sigma_{i=1}^m A_{ji}^2}$, since $\|O_j\|^2 = \frac{1}{2}\Sigma_{i=1}^m A_{ji}^2$, and $O_j \bullet Y^p = \Sigma_{i=1}^m A_{ji}\frac{(Y^p_{i+1\ 1} + Y^p_{1\ i+1})}{2}$.

The operation in Line 10

1: $Z_{n+1+i}^{p+1} = Y^p + (\frac{-C_i \bullet Y^p}{\|C_i\|^2})C_i$

can be replaced by

1: $Z_{n+1+i}^{p+1} = Y^p$
2: $Z_{n+1+i}^{p+1}[1, 2:m+1] = Z_{n+1+i}^{p+1}[1, 2:m+1] - \frac{c_i}{2}(\Sigma^{\frac{1}{2}T})_i$
3: $Z_{n+1+i}^{p+1}[1, m+1+i] = Z_{n+1+i}^{p+1}[1, m+1+i] + \frac{c_i}{2}$
4: $Z_{n+1+i}^{p+1}[2:m+1, 1] = Z_{n+1+i}^{p+1}[2:m+1, 1] - \frac{c_i}{2}(\Sigma^{\frac{1}{2}T})_i$
5: $Z_{n+1+i}^{p+1}[m+1+i, 1] = Z_{n+1+i}^{p+1}[m+1+i, 1] + \frac{c_i}{2}$,

with $c_i = \frac{-C_i \bullet Y^p}{\|C_i\|^2} = \frac{\Sigma_{k=1}^m (\Sigma^{\frac{1}{2}T})_{ik}(Y^p_{k+1\ 1} + Y^p_{1\ k+1}) - Y^p_{m+i+1\ 1} - Y^p_{1\ m+i+1}}{1 + \Sigma_{k=1}^m (\Sigma^{\frac{1}{2}T})_{ik}^2}$, since $\|C_i\|^2 = \frac{1}{2}(1 + \Sigma_{k=1}^m (\Sigma^{\frac{1}{2}T})_{ik}^2)$, and $C_i \bullet Y^p = -\Sigma_{k=1}^m (\Sigma^{\frac{1}{2}T})_{ik}\frac{Y^p_{k+1\ 1} + Y^p_{1\ k+1}}{2} + \frac{Y^p_{m+i+1\ 1} + Y^p_{1\ m+i+1}}{2}$.

The operation in Line 12

1: $Z_{n+m+2}^{p+1} = Y^p + (\frac{0 - Q \bullet Y^p}{\|Q\|^2})Q$

can be replaced by

1: $Z_{n+m+2}^{p+1} = Y^p$
2: $Z_{n+m+2}^{p+1}[m+1+i, m+1+i] = Z_{n+m+2}^{p+1}[m+1+i, m+1+i] + q$ for i between 1 and m.
3: $Z_{n+m+2}^{p+1}[2m+2, 2m+2] = Z_{n+m+2}^{p+1}[2m+2, 2m+2] - q$
4: $Z_{n+m+2}^{p+1}[2m+3, 2m+3] = Z_{n+m+2}^{p+1}[2m+3, 2m+3] + q$,

with $q = \frac{0 - Q \bullet Y^p}{\|Q\|^2} = \frac{\Sigma_{k=m+2}^{2m+1} Y^p_{kk} - Y^p_{2m+2\ 2m+2} + Y^p_{2m+3\ 2m+3}}{m+2}$, since $\|Q\|^2 = m+2$, and $Q \bullet Y^p = -\Sigma_{k=m+2}^{2m+1} Y^p_{kk} + Y^p_{2m+2\ 2m+2} - Y^p_{2m+3\ 2m+3}$.

The operation in Line 14

1: $Z_{n+m+1+i}^{p+1} = Y^p + (\frac{0 - D_i \bullet Y^p}{\|D_i\|^2})D_i$

can be replaced by

1: $Z_{n+m+1+i}^{p+1} = Y^p$
2: $Z_{n+m+1+i}^{p+1}[i, i] = Z_{n+m+1+i}^{p+1}[i, i] + d_i$
3: $Z_{n+m+1+i}^{p+1}[1, i] = Z_{n+m+1+i}^{p+1}[1, i] - \frac{d_i}{2}$
4: $Z_{n+m+1+i}^{p+1}[i, 1] = Z_{n+m+1+i}^{p+1}[i, 1] - \frac{d_i}{2}$,

with $d_i = \frac{0 - D_i \bullet Y^p}{\|D_i\|^2} = -\frac{2}{3}(Y^p[i,i] - \frac{Y^p[i,1] + Y^p[1,i]}{2})$, since $\|D_i\|^2 = \frac{3}{2}$, and $D_i \bullet Y^p = Y^p[i,i] - \frac{Y^p[i,1] + Y^p[1,i]}{2}$.

The operation in Line 16

1: $Z^{p+1}_{n+2m+3} = Y^p + (\frac{0-R\bullet Y^p}{\|R\|^2})R$

can be replaced by

1: $Z^{p+1}_{n+2m+3} = Y^p$
2: $Z^{p+1}_{n+2m+3}[2m+4, 2m+4] = Z^{p+1}_{n+2m+3}[2m+4, 2m+4] - l$
3: $Z^{p+1}_{n+2m+3}[1, 2m+2] = Z^{p+1}_{n+2m+3}[1, 2m+2] + \frac{l}{2}$
4: $Z^{p+1}_{n+2m+3}[2m+2, 1] = Z^{p+1}_{n+2m+3}[2m+2, 1] + \frac{l}{2}$,

with $l = \frac{0-R\bullet Y^p}{\|R\|^2} = \frac{2}{3}(Y^p_{2m+4\ 2m+4} - \frac{Y^p_{2m+2\ 1}+Y^p_{1\ 2m+2}}{2})$ since $\|L\|^2 = \frac{3}{2}$ and $L \bullet Y^p = -Y^p_{2m+4\ 2m+4} + \frac{Y^p_{2m+2\ 1}+Y^p_{1\ 2m+2}}{2}$.

These sparse computations are integrated in Algorithm 5 to solve the SDP problem (III.13) that gives a lower bound to validate the heuristic approach DFW. All this will be numerically tested in the next section.

III.2 Experimental results

The following experimental results aim at evaluating the quality of the proposed solution by DFW Algorithm by checking the gap obtained by the bidualization of Problem (II.6). For this, an important observation is that the bidual problem (III.13) is an SDP problem. In order to compute this gap, we test both CVXPY SDP solver and Pierra's algorithm.

First, we evaluate the quality of the solution of DFW Algorithm and analyze the obtained lower bound. Then, for the SDP relaxation, we compare solutions obtained by CVXPY and Pierra's algorithm, and we show the storage economy resulting from using Pierra's algorithm. This storage economy is especially due to the fact that we are taking advantage of the matrices sparsity in Problem (III.13).

III.2.1 Experimental setup

As in Chapter II, we consider the robust counterpart of the shortest path problem with an undirected grid graph for different sizes. Recall that for a grid graph $L \times L$, the number of nodes is $n = L^2$, and the number of edges is $m = 4L(L-1)$. For the definition of Problem (II.6), the random mean vector μ and the random covariance matrix Σ are chosen randomly and $\Omega = 1$, as in Section II.5.

The implementation of both the computation of DFW robust solutions and the CVXPY based solver are written using `Python 3.8.5`. However, Pierra's algorithm is implemented using `Matlab R2018b`. Computations have been performed

on the supercomputer facilities of Mésocenter de calcul de Franche-Comté in Besançon, France [1]. A global recap for the coding language information and the experimental parameters of this section can be found in Table III.2.

III.2.2 Numerical evaluation of the heuristic approach DFW

Here, we first show the solutions of Problem (II.6) proposed by DFW algorithm and CPLEX for problem sizes $L \in \{3, 4, \ldots, 10\}$. No surprises here, we have already seen in Chapter II that our proposed heuristic DFW gives the same solution as the optimal one for these problem sizes. The goal in this chapter anyhow is to develop a lower bound validation to avoid comparing with exact methods to evaluate the quality of the obtained solution. For this purpose, we compute the lower bound d^{**} of Problem (II.6). In this part, this lower bound is computed using CVXPY. Finally, we compute the relative biduality gap RBG (given in Definition (III.16)) which allows to evaluate the quality of the solution given by DFW algorithm, and the *performance ratio*, which is useful for comparison with other work.

For experiments with DFW algorithm, constant parameters are $\varepsilon = 10^{-6}$ and $K = 1000$. Table III.1 shows results for problem sizes $L \in \{3, 4, \ldots, 10\}$. First of all, we note that in all the cases processed, DFW algorithm gives the same solution as CPLEX. Second, concerning the relative gap, since we theoretically only have the weak duality for the problem we are solving, the biduality gap is not necessarily zero even if the solution is optimal. Thus, if the gap is small, it means that the heuristic solution is close to the optimal solution, but the opposite is false. Indeed a large gap does not mean that the heuristic solution is far from the optimal solution in the worst case. In all the processed cases, the lower bound d^{**} is less than the optimal solution p^*, which validates the developments and the computations. Next, the obtained relative gap RBG is between 0.1917 and 0.3178. This gap is a metric that allows to measure how far the heuristic approach is from the optimal solution. In other words, the heuristic solution is between 19.17% and 31.78% from optimality, in the worst case. In our case, the range of values for the *performance ratio* is: $0.7588 \leq \rho \leq 0.8391$. This is comparable to 0.87, the highest performance ratio obtained for the *Max-Cut* problem([Karloff 99]).

It would be interesting to test other cases of larger problems where the comparison with CPLEX is not possible, and check if the relative gap stays in the same interval as the processed cases. Indeed, in the processed cases, DFW algorithm gives the optimal solution, and thus the gap only comes from $p^* - d^{**}$ (see Equation (III.14)).

[1] http://meso.univ-fcomte.fr/

L	Solution of DFW $g(\hat{x})$	Optimal solution by CPLEX p^*	Lower bound by CVXPY d^{**}	Relative gap RBG	Performance ratio ρ
3	223.8807	223.8807	169.8902	0.3178	0.7588
4	302.9097	302.9097	230.64099	0.3133	0.7614
5	381.3647	381.3647	292.6109	0.3033	0.7673
6	498.444952	498.444952	401.92866	0.2401	0.8064
7	524.41995	524.41995	422.3119	0.2418	0.8053
8	625.46595	625.46595	524.83906	0.1917	0.8391
9	659.0601	659.0601	542.6984	0.2144	0.8234
10	604.0187	604.0187	492.4042	0.2267	0.8152

Table III.1: Comparison of the solution proposed by DFW with the optimal solution using CPLEX, and the lower bound given by CVXPY.

Now that the evaluation of solutions given by DFW algorithm is done using both CPLEX and the relative gap computed by CVXPY, an issue remains as discussed in Section III.1.3: CVXPY needs a huge amount of memory to store matrices. That has justified the use of an alternative approach with Pierra's algorithm using sparse computations detailed in Section III.1.3.c. In the next section, numerical results obtained using Pierra's algorithm are presented, as well as the resulting gain in memory storage.

III.2.3 Numerical results of Pierra's algorithm

In this part, we only consider the bidual problem (III.13). Despite that the goal of this chapter is to evaluate the quality of the solution of the DFW approach by computing a gap with the solution of a bidual problem, here, the focus is on the numerical results of solving only the bidual problem (III.13).

We show the results when using Pierra's algorithm for Problem (III.13) in comparison with the direct method implemented with CVXPY for problem sizes $L \in \{3, 4, \ldots, 10\}$. For these experiments, constant parameters are $\epsilon = 10^{-4}$, $\lambda = 0.5$, $k = 3$ and $\alpha = 10^{-8}$. Z^1 is chosen as $0_{(2m+4,2m+4)}$. Table III.3 shows the computation time and memory space needed for the computation using respectively CVXPY and Pierra's algorithm, as well as the percentage of optimality of Pierra's solution compared to CVXPY after about 10000 iterations. The memory space saving is important. For $L = 10$, we have saved from 3.45 GigaBytes to 26 MegaBytes: we have won a factor of 100. In a reasonable computation time, that is however longer than the computation time of CVXPY, we get great per-

Coding language/Parameter	Version/value
`Python`	`3.8.5`
`Matlab`	`R2018b`
Ω	1
L	$\{3, 4, \ldots, 10\}$
ε	10^{-6}
K	1 000
ϵ	10^{-4}
λ	0.5
k	3
α	10^{-8}
P	10 000

Table III.2: Information about the coding language and parameters.

centages from optimality. Figure III.1 shows an example of the evolution of the objective function along the iterations of Pierra's algorithm for the problem size $L = 10$, compared to the optimal solution obtained by CVXPY. In this example $P = 15000$, and $\epsilon = 10^{-4}$. A very good convergence can be observed at the last iterations shown in Table III.3: we are at 99.93% from optimality.

However, it has been observed for $L = 3$ that after the convergence, the algorithm diverges a bit from the optimal solution. This is shown in Figure III.2, it would be interesting to understand this behavior.

The next section contains a global discussion about these numerical experiments.

III.2.4 Discussion

As a synthesis of these numerical experiments, it is possible to make the following comments. The lower bounds using Pierra's algorithm have been provided for small problem sizes. Thus, the contribution of this chapter is to propose a method to evaluate the solution of a heuristic algorithm for Problem (II.6) without comparing it with CPLEX, but rather with a lower bound. For this, a challenge has been encountered, since it is well known that using the duality makes the problems easier but bigger, as the dual problem is usually polynomial, but has more variables and more constraints. This challenge has been tackled using Pierra's algorithm with the sparse computations. Here, the goal is twofold: first, put the algorithm proposed by Pierra in 1984 back in the spotlight for its efficiency even if it has not been used much. Indeed this algorithm has the potential to compete with existing methods such as interior-point methods [Nesterov 94] and ADMM [Boyd 11]. The

	Time(s)		Storage needed (mB)		
L	Direct CVXPY	Sparse Pierra	Direct CVXPY	Sparse Pierra	Optimality percentage of Pierra (% CVXPY)
3	11	3.7	1.29792	0.13632	96.4%
4	49.6761	97.2	9.2	0.50496	77%
5	145.93	631	40.45	1.358848	86%
6	394.2456	1005.4	132.88	3.008448	92.2%
7	935.8	2275	358.82	5.841792	92.4%
8	2274.85	7826	841.73	10.32448	96%
9	4724.6	22338	1776.192	16.99968	97%
10	9244.87	63585	3451.17	26.488128	99.93%

Table III.3: Comparison between a direct method using CVXPY versus the sparse version of Pierra's Algorithm.

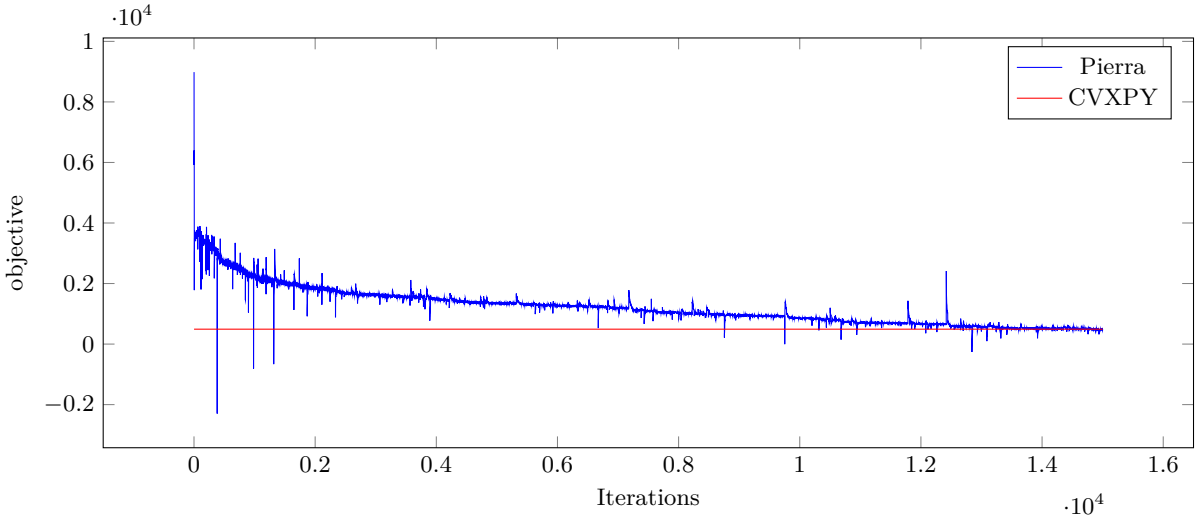


Figure III.1: Evolution of the objective function along 15 000 iterations in Pierra's Algorithm for $L = 10$ compared to CVXPY's implementation.

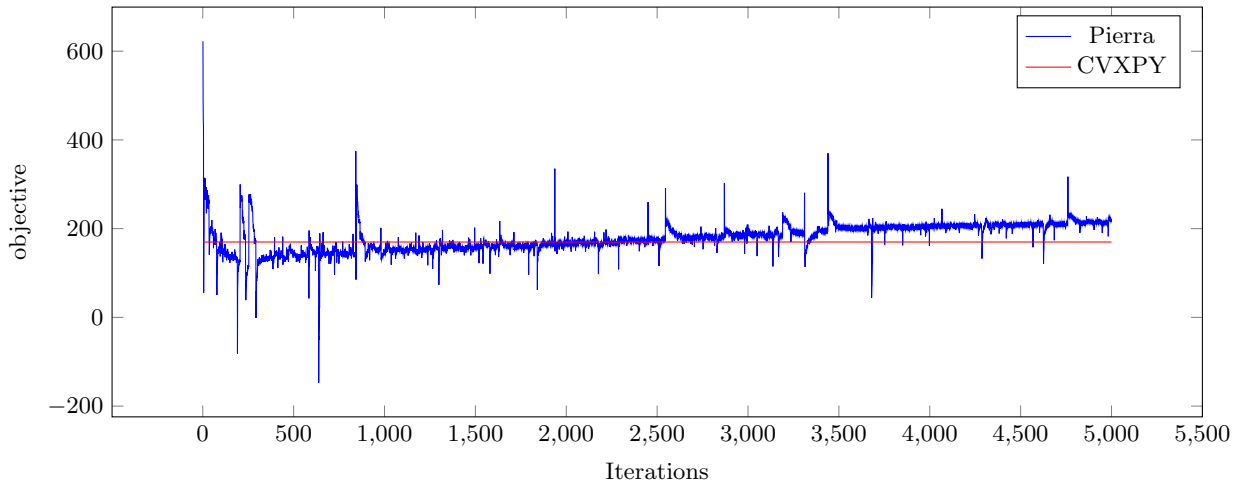


Figure III.2: Evolution of the objective function along 5 000 iterations in Pierra's Algorithm for $L = 3$ compared to CVXPY's implementation.

second goal is to show the power of having an explicit algorithm instead of a black box solver, since, due to that, the sparse computations were possible, reducing drastically the memory space necessity.

Before stating the limitations and perspectives of these numerical experiments, it is important to resume the different challenges of the validation part of the study. First, the comparison with CPLEX is not possible for grid graph with size L larger than 40. Thus, we proposed the validation with a lower bound, and our goal was to go further than $L = 40$. But the lower bound implied a storage challenge: for $L = 40$, the matrices storage necessitates 17.5 terabytes. We wanted to rise to this challenge by proposing the sparse version of Pierra's Algorithm, but this time, the new challenge is the speed: the algorithm starts to take a long time to converge for L greater than 10.

As a perspective, an acceleration for Pierra is needed. But in order to be able to do so, some challenges concerning Pierra's algorithm should be dealt with, such as the stopping criteria in Algorithm 5 Line 19 and the performance of the algorithm that has to be sped up. One should note that a convenient idea for acceleration is the parallelization. Indeed, the architecture of the algorithm allows a very easy parallelization, since the projections on each constraint space are independent (Algorithm 5 Lines 6 to 17). Thus, a parallel implementation (Algorithm 5 Line 18) could speed up the algorithm.

III.2.5 Difficulties in the experiments

This section describes the numerical difficulties we went through before obtaining good results. The first difficulty is due to the bidualization of Problem (II.6), since we intended to use a simplified quadratic form of Problem (II.6). We used Pierra's Algorithm to solve it, but we struggled to obtain convergence for this form. This part is explained below in Section III.2.5.a. The second difficulty was encountered in the numerical validation of the sparse computations by comparing them with the full ones. This is described later in Section III.2.5.b.

III.2.5.a About the convergence of a simplified quadratic form of Problem (II.6)

To get a lower bound for Problem (II.6), we intended to write Problem (II.6) in a simplified quadratic form as the following:

$$\min_{\substack{x\in X, z\in\mathbb{R}^+ \\ x^T\Sigma x\leq z^2}} \mu^T x + z. \tag{III.31}$$

In comparison with the quadratic form (III.9), we simplified $y^T y \leq z^2$ and $y = (\Sigma^{\frac{1}{2}})^T x$ into one constraint $x^T\Sigma x \leq z^2$. The idea was to bidualize this form ((III.31)), because its bidual can reduce from $(2m+4)^2$ to $(m+4)^2$ variables and from $n+2m+3$ to $n+m+3$ constraints in comparison with the bidual (III.13) of Problem (III.9) Indeed, the bidual of Problem (III.31) has the following form:

$$\begin{aligned}
&\min M \bullet Z \\
&\text{s.t. } Z \in S_{m+4} \\
&O_j \bullet Z = b_j, j = 1, \ldots, n, \\
&Q \bullet Z = 0, \\
&D_i \bullet Z = 0, \quad i = 2, \ldots, m+1 \\
&R \bullet Z = 0, \\
&Z \succeq 0.
\end{aligned} \tag{III.32}$$

The problem is when trying to solve Problem (III.32) using the algorithm of Pierra with different parameters, it was hard to get the convergence. The best parameters for $L = 3$ were $\alpha = 10^{-5}$, $k = 3$, $\lambda = 0.5$, and after 8 000 000 iterations, the objective function evaluated at the last iteration Z_{Final} equaled $M \bullet Z_{\text{Final}} = 225.6$. Whereas the solution of CVXPY Z_{CVXPY} corresponded to $M \bullet Z_{\text{CVXPY}} = 163.355$. On the other side, when we used Pierra's algorithm to solve (III.13), in less than 2 000 000 iterations, it converged, giving $M \bullet Z_{\text{Final}} = 159.88$. In order to understand more what is happening, displaying the different projections shows if the algorithm converges or not, because the solution should verify the constraints. The final solution Z should satisfy $|A \bullet Z - b|$ close to zero for every constraint in

the form $A \bullet Z = b$, and the eigenvalues of Z should almost be positive, for the constraint $Z > 0$. So when we displayed the projections, all of them were around 10^{-2} and 10^{-3}, except for Q, $Q \bullet Z$ was around 200. The explanation that we found is that it is an important constraint in the problem, and it is only represented by one constraint. Whereas, in Problem (III.13), the same constraint is expressed by $m + 1$ constraints, which means that its corresponding projection has more representation in the mean of projections in Pierra's Algorithm (Algorithm 5 Line 18). Another explanation is the poor conditioning of the Problem (III.32), which means that in applied mathematics, it is known that some algorithms can work well for a form of problem than another, even if these two forms correspond to the same problem, and probably, this happened for Problems (III.13) and (III.32) .

III.2.5.b Sparse versus full computations

In Pierra's algorithm 5 that we used to solve (III.13), recall that mathematical operations that include matrix multiplications, additions and subtractions are needed, and they are detailed in Section III.1.3.c. Here, we show the difficulty we had when validating the computation in Algorithm 5.

In the implementation stage, in order to validate the sparse version of Pierra's Algorithm, a comparison with the full version was made. Unfortunately, we did not obtain the same results in the comparison step. After months of investigation, and even also changing the coding language from Python to Matlab, we have not found any explanation.

Finally, we discovered that the difference between the solutions of Algorithm 5 between the sparse and the full version was not due to any error in computations, but it was due to the sensitivity of Algorithm 5 to numerical precision: even implementing the dot product in two equivalent ways could change the final result of Algorithm 5.

The analysis we have about this is that this sensitivity comes probably from Line 22 of Algorithm 5, since as we can see, β^{p+1} has Den $= \|B'^{p+1} - Z^p\|^2$ in its denominator. If the term Den equals 10^{-15} in one version, and 10^{-17} in another, then, there is 10^{-2} of difference in the denominator, and thus, there is a difference of 100 in β. Not to forget that the denominator term is square, and that the difference in precision builds up along the iterations, and thus the difference becomes bigger and bigger along the iterations.

III.2.6 Synthesis

To sum up the numerical experiments, the difficulties in Section III.2.5 have been solved, and the limitations in the discussion (Section III.2.4) open to perspectives that are promising for the method proposed in this chapter. Despite the difficulties and challenges, this chapter proposes a polynomial time evaluation of the quality of the solution of DFW heuristic approach without having the memory storage issue of the bidual problem.

In this chapter, and in the previous one (Chapter II), all the work has been applied on the robust shortest path problem. The next chapter studies another problem that is more difficult, but more interesting in its applications: the robust `k`-median clustering problem.

Chapter IV

A second heuristic approach based on Frank-Wolfe for the k-median clustering problem

This chapter proposes a Frank-Wolfe based approach for the k-median clustering. First, the classical k-median problem is written as a binary linear programming problem (BLP) by a matrix flattening step. Then, the robust k-median clustering problem under ellipsoidal uncertainty is written as a binary non-linear Problem. The Frank-Wolfe based approach to solve this binary non-linear problem consists in relaxing the binarity constraints and using the Frank-Wolfe Algorithm with a rounding technique for the mean of the intermediate steps of the algorithm. Results show that this approach gives the optimal solution in most of the cases, and that it gives close-to-optimal solutions when they are not optimal.

IV.1 Motivation and context

k-median clustering is a classical method of unsupervised learning that aims to partition a number of measurements in k-clusters. Unsupervised learning gets increasing attention with the raise of machine learning in all domains. Thus, studies about this problem are applied in many domains. The issue we are interested in is how to deal with the uncertainty in this problem. Indeed, due to measurements errors, it is important to put the robust optimization expertise in the service of this important clustering problem. If the input data of the clustering problem are subject to uncertainties, which are the points that we want to group in clusters, we want to determine a robust counterpart of the k-median problem in the case of ellipsoidal uncertainty. The choice of the ellipsoidal uncertainty set is motivated by the fact that it is important to take in account the correlations between the distances between the different points of the data set. Up to our knowledge, robust clustering is a relatively new topic. Related works have been done in [Bургard 15, Li 16] for robust k-median and k-means clustering with ellipsoidal and budgeted uncertainty. A very interesting paper by Bertsimas *et al.* about robust classification that presents robust support vector machines (SVM), logistic regression, and decision trees can be found in [Bertsimas 19]. Next, works with a focus on robust SVM can be found in [Singla 20, Pant 11, Trafalis 07]. Finally, a book about robust data mining is the one of Xanthopoulos *et al.* [Xanthopoulos 12]. Up to our knowledge, there does not exist any work on robust k-median clustering with ellipsoidal uncertainty set.

The developments with all the details can be found in the following.

IV.2 Problem formulation

In this section, we consider a robustification of the k-median clustering problem. Let $P = \{p_1, \dots, p_n\}$ a set of n points. The k-median problem consists in choosing, among the points in P, k clusters that minimize the sum of the distances between the points $p \in P$ and their cluster centers. It can be expressed in the form of a binary programming problem, as formulated by Awasthi *et al.* in [Awasthi 15]. The formulation is the following:

$$\begin{aligned} \min_{(z_{ij})_{ij\in\{1,\dots,n\}^2}\in\mathbb{R}^{\{1,\dots,n\}^2}} & \Sigma_{i,j\in\{1,\dots,n\}^2} d(p_i,p_j) z_{ij} \\ \text{s.t. } & \Sigma_{i\in\{1,\dots,n\}} z_{ij} = 1 \quad \forall j \in \{1,\dots,n\} \\ & z_{ij} \leq y_j \quad \forall i,j \in \{1,\dots,n\}^2 \\ & \Sigma_{i\in\{1,\dots,n\}} y_i = \mathtt{k} \\ & z_{ij}, y_i \in \{0,1\}, \end{aligned} \tag{IV.1}$$

where $d(p_i, p_j)$ is the real positive distance between the points p_i and p_j, y_i indicates whether the point p_i is a cluster center or not, z_{ij} tells us whether p_j is assigned to p_i as center or not. The constraints of Problem(IV.1) assure that each point is assigned to one and only one cluster center, that we do not assign a point to another one unless the second is a center and that there exist k centers. The real positive distance between two points can represent any positive value in any metric space. This problem can also be written using a matrix form:

$$\begin{aligned} \min_{Z\in\{0,1\}^{n\times n}} & D \bullet Z \\ \text{s.t. } & \Sigma_{i=1}^{n} Z_{ij} = 1 \quad \forall j \in \{1,\dots,n\} \\ & Z_{ij} \leq Z_{ii} \quad \forall i,j \in \{1,\dots,n\}^2 \\ & \Sigma_{i=1}^{n} Z_{ii} = \mathtt{k}, \end{aligned} \tag{IV.2}$$

where $\bullet$ is the same inner product defined in the previous chapter, D is a symmetric matrix with dimension $n \times n$, with D_{ij} being the distance between p_i and p_j for $i \in \{1,\dots,n\}$, $j \in \{1,\dots,n\}$. The matrix D is symmetric and its diagonal has zero entries, since $d(p_i,p_j) = d(p_j,p_i)$, and $d(p_i,p_i) = 0$ for i and j belonging to $\{1,\dots,n\}$. Z is the matrix of variables with dimension $n \times n$, where the element Z_{ij} is $(z_{ij})_{ij\in\{1,\dots,n\}^2}$ for $i \neq j$, and with Z_{ii} is $(y_i)_{i\in\{1,\dots,n\}}$. The constraints in this second form (Problem (IV.2)) are translated as the following: Z is a binary matrix in which the sum over each column equals 1, the non-diagonal elements of each row in Z are less or equal than the diagonal value of this row. Finally, the sum over the diagonal of Z equals the number of cluster centers k. In real life applications, the distances between the points defined above are subject to

uncertainties. Thus a robust clustering solution seems mandatory. The aim of the following is to determine a robust counterpart of the k-median problem in the case of ellipsoidal uncertainty. To do so, we write Problem (IV.1) in the formulation (II.1), which is, we recall,

$$\min_{x \in X} c^T x.$$

The first step is to consider the flattened version z (respectively d) with length n^2 of Z (respectively D) of Problem (IV.2), then the deterministic k-median problem can be written as the following form

$$\begin{aligned} \min d^T z \\ \text{s.t. } \Sigma_{i=1}^n z_{n(i-1)+j} = 1 \quad \forall j \in \{1, \dots, n\} \\ z_{n(i-1)+j} \leq z_{n(i-1)+i} \quad \forall i, j \in \{1, \dots, n\}^2 \\ \Sigma_{i=1}^n z_{n(i-1)+i} = \mathtt{k} \\ z \in \{0,1\}^{n^2}. \end{aligned} \tag{IV.3}$$

The flattening is done observing that, in Problem (IV.2), an element Z_{ij} with $i \in \{1, \cdots, n\}$ and $j \in \{1, \cdots, n\}$ is represented in Problem (IV.3) as $z_{n(i-1)+j}$. Then, this problem has the formulation (II.1) which can be written as

$$\min_{z \in X} d^T z, \tag{IV.4}$$

with $X \subseteq \{0,1\}^{n^2}$ and where

$$\begin{aligned} X = \Big\{ z \in \{0,1\}^{n^2} \text{ s.t.} \\ \Sigma_{i=1}^n z_{n(i-1)+j} = 1 \quad \forall j \in \{1, \dots, n\}, \\ z_{n(i-1)+j} \leq z_{n(i-1)+i} \forall i, j \in \{1, \dots, n\}^2, \\ \Sigma_{i=1}^n z_{n(i-1)+i} = \mathtt{k} \Big\}. \end{aligned}$$

If we suppose that the distances between the points are uncertain, and making the assumption that these distances follow a multinormal distribution, then the vector d also follows a multinormal distribution. If we note the expectation of d by $\mu \in \mathbb{R}^{n^2}$ and the covariance matrix by $\Sigma \in \mathbb{R}^{n^2 \times n^2}$, then by following the development done previously in Section II.2, the robust clustering problem is reduced to solving the following non-deterministic problem:

$$\min_{z \in X} \mu^T z + \Omega \sqrt{z^T \Sigma z}.$$

By replacing Σ by $\Omega^2 \Sigma$, we obtain

$$\min_{z \in X} \mu^T z + \sqrt{z^T \Sigma z}, \tag{IV.5}$$

with

$$X = \Big\{ z \in \{0,1\}^{n^2} \text{ s.t.}$$
$$\Sigma_{i=1}^{n} z_{n(i-1)+j} = 1 \quad \forall j \in \{1,\dots,n\},$$
$$z_{n(i-1)+j} \leq z_{n(i-1)+i} \forall i,j \in \{1,\dots,n\}^2,$$
$$\Sigma_{i=1}^{n} z_{n(i-1)+i} = \mathtt{k} \Big\}.$$

The missing ingredient to tackle the problem is to describe the uncertainty modelling of the vector d in Problem (IV.4) in function of the uncertainty modelling of the distances between the points in Problem (IV.1). This is done due to Theorem 1.

Theorem 1. *Let $v \in \mathbb{R}^{\frac{n(n-1)}{2}}$ a random variable that has a multinormal distribution $\mathcal{N}(\mu_v, \Sigma_v)$. If we define the vector $d \in \mathbb{R}^{n^2}$ as the flattened version of the matrix*

$$\begin{bmatrix} 0 & v_1 & v_2 & \dots & v_{n-1} \\ v_1 & 0 & v_n & \dots & v_{2n-3} \\ v_2 & & & & \\ \vdots & & & & \\ & & & \ddots & v_{\frac{n(n-1)}{2}} \\ v_{n-1} & \dots & & v_{\frac{n(n-1)}{2}} & 0 \end{bmatrix},$$

which means that

$$d = \begin{bmatrix} 0 & v_1 & v_2 & \dots & v_{n-1} & v_1 & 0 & v_n & \dots & v_{2n-3} & \dots & v_{n-1} & \dots v_{\frac{n(n-1)}{2}} & 0 \end{bmatrix},$$

then d has a multinormal distribution $\mathcal{N}(\mu, \Sigma)$, with $\mu = L\mu_v$, and $\Sigma = L\Sigma_v L^T$, where L is a matrix with dimension $n^2 \times \frac{n(n-1)}{2}$ such that $d = Lv$.

Proof. First, $d = Lv$ has a multinormal distribution, since it is a linear transformation of v, which is a random variable that has a multinormal distribution [Gut , Definition 3.1]. Second, $\mu = L\mu_v$. Indeed,

$$\begin{aligned} \mu &= \mathbb{E}[d] \\ &= \mathbb{E}[Lv] \\ &= L\mathbb{E}[v] \\ &= L\mu_v. \end{aligned}$$

Finally, $\Sigma = L\Sigma_v L^T$. Indeed,

$$\begin{aligned} \Sigma &= \mathbb{E}[(d-\mu)(d-\mu)^T] \\ &= \mathbb{E}[(Lv - L\mu_v)(Lv - L\mu_v)^T]. \end{aligned}$$

This holds since $d = Lv$ and $\mu = L\mu_v$. So,

$$\begin{aligned}\Sigma &= \mathbb{E}[(L(v-\mu_v))(L(v-\mu_v))^T] \\ &= \mathbb{E}[L(v-\mu_v)(v-\mu_v)^T L^T] \\ &= L\mathbb{E}[(v-\mu_v)(v-\mu_v)^T]L^T \\ &= L\Sigma_v L^T.\end{aligned}$$

□

Theorem 1 implies that the uncertainty in the distances between the points in the k-median problem (IV.1) can be modelled by an uncertainty in the vector d in(IV.3). Indeed, the multinormality of the distances between the points that is described by the vector v in Theorem 1 implies the multinormality of the vector d, and after that, we can work on the ellipsoidal uncertainty over the vector d. This theorem is used in the experimental setup, for the generation of adequate mean vector μ and covariance matrix Σ. The goal of the following is to solve the robust k-median problem under ellipsoidal uncertainty by solving Problem (IV.5). In the rest of this chapter, it is common to do the confusion between the flattened version and the matrix version of the solutions z, as they are simply two different representations of z, and the transformation between them is easy.

IV.3 Problem illustration

In this section, we find it useful to consider a simple example to illustrate the costs, the variables and the solutions in the different writings (IV.2) and (IV.4) of the problem. Let $P = \{p_1, \ldots, p_{10}\}$ a set of 10 points such that the distances between these points are represented in the matrix D with dimension 10×10:

$$D = \begin{bmatrix} 0 & 2 & 11 & 4 & 5 & 10 & 3 & 5 & 13 & 11 \\ 2 & 0 & 10 & 2 & 3 & 9 & 2 & 4 & 12 & 9 \\ 11 & 10 & 0 & 9 & 7 & 2 & 8 & 6 & 2 & 3 \\ 4 & 2 & 9 & 0 & 2 & 8 & 3 & 4 & 11 & 8 \\ 5 & 3 & 7 & 2 & 0 & 7 & 3 & 3 & 9 & 6 \\ 10 & 9 & 2 & 8 & 7 & 0 & 7 & 4 & 5 & 5 \\ 3 & 2 & 8 & 3 & 3 & 7 & 0 & 2 & 10 & 8 \\ 5 & 4 & 6 & 4 & 3 & 4 & 2 & 0 & 9 & 6 \\ 13 & 12 & 2 & 11 & 9 & 5 & 10 & 9 & 0 & 3 \\ 11 & 9 & 3 & 8 & 6 & 5 & 8 & 6 & 3 & 0 \end{bmatrix},$$

where $D[i, j]$ is the distance between p_i and p_j. For example $D[1, 2] = d(p_1, p_2) = 2$. The matrix D is indeed symmetric and its diagonal has zero entries. For this input set of points, the 2-median problem consists in finding 2 clusters that minimize

the sum of the distances between the points in $P = \{p_1, \ldots, p_{10}\}$ and their cluster centers. It is the problem (IV.2), with $n = 10$, $\mathtt{k} = 2$, and the cost function represented in the matrix D. An example of a feasible solution is illustrated in Figure IV.1. It can be represented by the following binary matrix Z:

$$Z = \begin{bmatrix} \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & \mathbf{1} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} \end{bmatrix}.$$

The diagonal of Z is enhanced in bold, since the information on the diagonal gives us the cluster centers. The cluster centers are p_3 and p_7 since $Z[3,3] = y_3 = 1$ and $Z[7,7] = y_7 = 1$. The non-diagonal entries of the row 3 (resp. 7) of the matrix Z with values equal to 1 correspond to the points that are associated with the cluster center p_3 (resp. p_7). For example, $Z[3,6] = z_{3\,6} = 1$ means that Point p_6 is associated with the cluster center p_3 . In total, Points p_6, p_9 and p_{10} are associated with the cluster center p_3, and the points p_1, p_2, p_4, p_5 and p_8 are associated with the cluster center p_7. The solution represented by the matrix Z satisfies the constraints of Problem IV.1:

- $\Sigma_{i=1}^{10} Z[i,j] = 1 \quad \forall j \in \{1, \ldots, 10\}$ (the sum over each column equals 1) reads: every point is associated with one and only cluster center,
- $Z[i,j] \leq Z[i,i] \quad \forall i,j \in \{1, \ldots, 10\}^2$ (the values of the non-diagonal entries are less or equal than the diagonal entry of the same row) reads: it is not possible to associate a point to a second one unless the second is a cluster center,
- $\Sigma_{i=1}^{10} Z[i,i] = 2$ (the sum over the diagonal equals the number of cluster centers) reads: The solution has exactly 2 cluster centers.

In addition, this feasible solution is the optimal one with the cost equals $D \bullet Z = (2+2+3)+(3+2+3+3+2) = 20$, and it is illustrated in Figure IV.1. Concerning the second representation (IV.4) of Problem (IV.1), it is obtained by a flattening of the matrix D (resp. Z) into a vector d (resp. z) of length $10 \times 10 = 100$, which is given by

$$d = \begin{bmatrix} 0 & 2 & 11 & 4 & 5 & 10 & 3 & 5 & 13 & 11 & 2 & 0 & \ldots & 9 & 11 & \ldots & 0 \end{bmatrix},$$

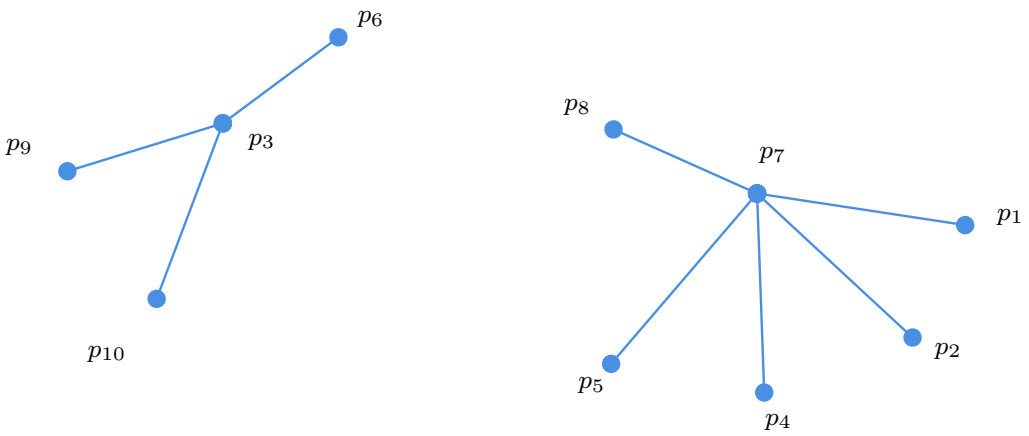


Figure IV.1: Simple example of a two cluster solution of a k-median problem for 10 points.

As mentioned in the previous section, the flattening is done the following way

$$\begin{aligned} D[i,j] &\to d_{10(i-1)+j} \quad \forall i,j \in \{1,\dots,10\}^2 \\ D[2,1] &\to d_{10(2-1)+1} = d_{11} \\ D[2,10] &\to d_{10(2-1)+10} = d_{20}. \end{aligned}$$

The first row of D forms the 10 first elements of d. Then the values of the elements $d_{11},\dots,d_{20}$ come from the second row of D, and so on. The same for the flattening of the matrix Z into the vector z. The next section comes back to our goal which is to solve the robust k-median problem under ellipsoidal uncertainty.

IV.4 A Frank-Wolfe based approach MFW for the k-median clustering

The goal now is to solve the robust k-median problem under ellipsoidal uncertainty. First, we show that it is not possible to apply DFW Algorithm on the robust k-median problem in hand, then we propose another algorithm based on the classical FW Algorithm. In MFW Algorithm and references to DFW Algorithm, we use in this chapter $\mathring{k}$ as an index for the algorithm's iterations instead of k to avoid confusion with the number of cluster centers k.

IV.4.1 Assumptions for DFW Algorithm not satisfied

Section IV.2 shows that this consists in solving a binary non-linear Problem (Problem (IV.5)), that is, we recall:

$$\min_{z\in X} \mu^T z + \sqrt{z^T \Sigma z}, \tag{IV.6}$$

with

$$X = \Big\{ z \in \{0,1\}^{n^2} \text{ s.t.}$$
$$\Sigma_{i=1}^{n} z_{n(i-1)+j} = 1 \quad \forall j \in \{1,\dots,n\},$$
$$z_{n(i-1)+j} \leq z_{n(i-1)+i} \forall i,j \in \{1,\dots,n\}^2,$$
$$\Sigma_{i=1}^{n} z_{n(i-1)+i} = \mathtt{k} \Big\}.$$

Problem (IV.6) is an NP-hard problem [Bertsimas 04b], but exact methods exist to solve the Binary Second Order Cone formulation (BSOCP) of the problem that is, we recall:

$$\begin{aligned} &\min \mu^T z + u \\ &\text{s.t. } (y,u)^T \in K_{n^2+1} \\ &y = (\Sigma^{\frac{1}{2}})^T z \\ &z \in X, y \in \mathbb{R}^{n^2}, u \in \mathbb{R}_+, \end{aligned} \tag{IV.7}$$

with $K_{n^2+1} = \Big\{ z \in \mathbb{R}^{n^2+1}; \|(z_1,\dots,z_{n^2})^T\|_2 \leq z_{n^2+1} \Big\}$ being a second order cone. This can be solved using the BSOCP solver of CPLEX [Manual 87], that is based on branch-and-bound methods. This follows the developpments done in Chapter II for the robust counterpart of general BLP problems under ellipsoidal uncertainty (see Section II.3 for more details). As also mentioned in Section II.3, the processing time of branch-and-bound methods for problems of large size becomes considerable. The goal of this part is to propose a heuristic algorithm for the robust k-median problem (IV.6). An algorithm named DFW has been proposed in Chapter II for the robust counterpart of a class of BLP problems. Recall that this algorithm is based on the idea of exploiting the direction steps of the classical FW Algorithm, that are binary solutions due to some assumptions that are linked with the exact binarity relaxation. Despite that the k-median problem has a BLP formulation (Problem (IV.4)), unfortunately, it is not possible to solve its robust counterpart (IV.6) using DFW Algorithm, since the assumptions in Section II.4.1 are not satisfied. In fact, Assumption **(A3)** is satisfied, since $0_{\{0,1\}^{n^2}}$ is not a clustering solution, but assumptions **(A1)** and **(A2)** are not satisfied. Indeed, let us recall the two assumptions **(A1)** and **(A2)**:

> **(A1)** For any real-valued vector a (not necessarily with positive entries), there exists an efficient algorithm to solve $\min_{z \in X} a^T z$,
>
> **(A2)** For any real-valued vector a, there exists a solution for $\min_{z \in \text{Conv}(X)} a^T z$ that belongs to X, where $\text{Conv}(X) \subset \mathbb{R}^m$ is the convex hull of X.

These assumptions are needed for the applicability of Algorithm DFW, and more precisely for the feasibility of the intermediate steps $s_{\mathring{k}}$ in iteration $\mathring{k}$ in Line 8 of Algorithm 2:

$$s^{(\mathring{k})} \in \underset{y \in \text{Conv}(X)}{\text{argmin}} \nabla g(x^{(\mathring{k})})^T y, \text{ with } s^{(\mathring{k})} \in X. \tag{IV.8}$$

For every iteration $\mathring{k}$, the condition $s^{(\mathring{k})} \in X$ is necessary, for its feasibility in the binary problem we are interested in solving (Problem (IV.6)). In addition, it is necessary for $s^{(\mathring{k})}$ to minimize $\nabla g(x^{(\mathring{k})})^T y$ over $\text{Conv}(X)$, for the convergence of $x^{(\mathring{k})}$ in $\text{Conv}(X)$. Noticing that the convex relaxation $\text{Conv}(X)$ of X is

$$\begin{aligned} \text{Conv}(X) = \Big\{ z \in [0,1]^{n^2} \text{ s.t.} \\ \Sigma_{i=1}^{n} z_{n(i-1)+j} = 1 \quad \forall j \in \{1,\dots,n\}, \\ z_{n(i-1)+j} \leq z_{n(i-1)+i} \forall i,j \in \{1,\dots,n\}^2, \\ \Sigma_{i=1}^{n} z_{n(i-1)+i} = \mathtt{k} \Big\}, \end{aligned} \tag{IV.9}$$

which consists simply in replacing $\{0,1\}^{n^2}$ by $[0,1]^{n^2}$, then in our case, these assumptions require the exact binarity relaxation condition for Problem (IV.4) for any real vector d. Unfortunately, this is not the case. Problem (IV.4) is a binary programming problem for which the convex relaxation is not exact, but only gives a lower bound for the problem.

IV.4.2 The proposed approach

Since we showed in the previous section that the assumptions needed to apply DFW Algorithm are not satisfied, instead, we propose here another adaptation of the classical FW algorithm to solve heuristically the robust k-median problem under ellipsoidal uncertainty (IV.5). The algorithm is stated in Algorithm 6, named MFW referring to Mean Frank-Wolfe. In Line 14 of Algorithm 6, the feasible round is detailed in Algorithm 7. The idea of the approach is the following: As in DFW Algorithm from Chapter II, the power of the algorithm is the exploration power of the iterates: while $x^{(\mathring{k})}$ converges to the optimal solution in $\text{Conv}(X)$, $s^{(\mathring{k})}$ is the optimal solution of the linear approximation of the objective function g locally around $x^{(\mathring{k})}$ for each iteration $\mathring{k}$. The problem to solve to obtain $s^{(\mathring{k})}$ in Line 8 of Algorithm 6 has a polynomial complexity, since it is a linear continuous problem. Unfortunately, unlike the shortest path problem, the k-median problem does not give any guaranty that the solution $s^{(\mathring{k})}$ is binary. The values of $s^{(\mathring{k})}$ belong to $\text{Conv}(X)$ defined in (IV.9). In DFW Algorithm, we made use of the information of all the $s^{(\mathring{k})}$ for all the iterations $\mathring{k}$ by taking the best $s^{(\mathring{k})}$ among the others. Here, the best solution is not binary, but rather takes values between 0 and 1. Nevertheless, the values between 0 and 1 could be interpreted on one side as a percentage of being a cluster center for the diagonal values (in the flattened version, these are the values in the indices $n(i-1)+i$). On the other side, they can be interpreted as a percentage of belonging to a cluster center for the non-diagonal values (in the flattened version, these are the values in the indices $n(i-1)+j$, with $i \neq j$). Thus, if we denote $\mu_{\mathring{k}-1}$ the mean $\frac{\Sigma_{i=1}^{\mathring{k}-1} s^{(i)}}{\mathring{k}-1}$ of the solutions $s^{(i)}$ until iteration $\mathring{k}-1$,

Algorithm 6 MFW: a Frank-Wolfe based algorithm to solve Problem (IV.6)

$x^{(0)} \in \text{Conv}(X)$ a random solution, $\varepsilon > 0$ close to zero, $\mathring{K}$ a maximum number of iterations.
$\mathring{k} \leftarrow 1$
stop $\leftarrow$ false
while $\mathring{k} \leq \mathring{K}$ and $\neg$stop **do**
  if $g(x^{(\mathring{k}-1)}) - g(x^{(\mathring{k})}) < \varepsilon$: **then**
    stop $\leftarrow$ true
  else
    $s^{(\mathring{k})} \in \underset{y \in \text{Conv}(X)}{\text{argmin}} \nabla g(x^{(\mathring{k})})^T y$
    $\gamma^{(\mathring{k})} \leftarrow \underset{\alpha \in [0,1]}{\text{argmin}}\, g(x^{(\mathring{k})} + \alpha(s^{(\mathring{k})} - x^{(\mathring{k})}))$
    $x^{(\mathring{k}+1)} \leftarrow x^{(\mathring{k})} + \gamma^{(\mathring{k})}(s^{(\mathring{k})} - x^{(\mathring{k})})$
  end if
  $\mathring{k}++$
end while
return a feasible round of $\mu_{\mathring{k}-1} = \dfrac{\Sigma_{i=1}^{\mathring{k}-1} s^{(i)}}{\mathring{k}-1}$

Algorithm 7 An algorithm for a feasible rounding of a solution in Conv(X)

Input: $s \in \text{Conv}(X)$, Output $s_f \in X$
Reshape s that is a vector in $[0,1]^{n^2}$ as a matrix with dimension $n \times n$
Sort the diagonal elements of s, and choose the $\mathtt{k}$ biggest elements. If we note $i_1, \ldots, i_\mathtt{k}$ the indices of these $\mathtt{k}$ biggest elements ($i_1 < i_2 < \ldots < i_\mathtt{k}$), then p_{i_1}, $\ldots$ and $p_{i_\mathtt{k}}$ are the chosen cluster centers $\rightarrow$ the diagonal elements of s_f are equal to 1 in the indices $i_1, \ldots, i_\mathtt{k}$, and 0 elsewhere
Keep the rows $i_1, \ldots, i_\mathtt{k}$ of s in a reduced matrix of dimension $\mathtt{k} \times n$, sort each column i of this reduced matrix, and associate the point p_i to the cluster center with the index l of the biggest value in the sorting of column $i \rightarrow$ in every column i of s_f, the value in row l equals 1, and the other values are 0
Reshape s_f that is a matrix with dimension $n \times n$ in a vector in $\{0,1\}^{n^2}$

where $\mathring{k}-1$ in Algorithm 6 is the last iteration at which the algorithm stops, this mean vector $\mu_{\mathring{k}-1}$ contains the information about the percentages for all the $s^{(i)}$. Thus, if, for instance, a point has not been affected as a cluster center in any of the $s^{(i)}$ with any positive percentage, then in the mean value $\mu_{\mathring{k}-1}$, it interferes with 0 percentage. Contrarily, if a point has been affected as a cluster center with big percentages in many iterations, it also corresponds to a big percentage in $\mu_{\mathring{k}-1}$. In addition to that, the advantage about the mean is that it also belongs to $\mathrm{Conv}(X)$ (i.e. $s^{(i)} \in \mathrm{Conv}(X)\ \forall i = 1, \ldots, \mathring{k}-1 \implies \mu_{\mathring{k}-1} \in \mathrm{Conv}(X)$). The last step consists in rounding $\mu_{\mathring{k}-1}$ to become binary, without violating the constraints of $\mathrm{Conv}(X)$. This is what we call in the following a feasible round. The Algorithm 7 takes a solution in $\mathrm{Conv}(X)$, and rounds it. This rounded solution then belongs to X. The proposed approach uses Algorithm 7 to round $\mu_{\mathring{k}-1}$. The idea of the feasible rounding algorithm is the following. Let s be a vector in $\mathrm{Conv}(X)$, and let s_f be the rounding of s, that is the output of the rounding algorithm. We start by reshaping s in a matrix with dimension $n \times n$ to facilitate the elements indexing, and we construct s_f in a matrix and then we flatten it. The feasible rounding is composed of two steps:

- The first step in the rounding process is to choose the cluster centers. Since it is required to have exactly `k` cluster centers, then the rounded solution has exactly `k` values in the diagonal that are equal to 1 and $n - \mathtt{k}$ values that are equal to 0. The algorithm rounds the `k` biggest values of the diagonal to 1, and the others to 0. This composes the diagonal elements of s_f.
- The next step is to fill the non-diagonal elements. We know that the only non-zero rows in s_f are the ones with the indices of the `k` biggest values, that are noted as $i_1, \ldots, i_{\mathtt{k}}$ in Algorithm 7, since the non-diagonal elements are less or equal than the diagonal element of the same row. So, we only need to fill the rows with the indices $i_1, \ldots, i_{\mathtt{k}}$. For every column i, there is only one value that equals 1, and the others are equal to 0: we choose to assign the point p_i to the cluster center p_l that corresponds to the biggest value in the i-th column of s ($s(l, i)$): this is the value that equals 1 in the i-th column of s_f.

This completes the matrix that will be flattened to give s_f. In order to illustrate the feasible rounding algorithm (Algorithm 7), we take the example of an input $s \in \text{Conv}(X)$ shaped in a matrix with dimension 10×10 defined as

$$s = \begin{bmatrix} 0.528 & 0 & 0.528 & 0 & 0 & 0 & 0 & 0.048 & 0 & 0.528 \\ 0 & 0.144 & 0.144 & 0.144 & 0.015 & 0 & 0.144 & 0.144 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.384 & 0 & 0.384 & 0.384 & 0.384 & 0 & 0 & 0.319 & 0.25 \\ 0 & 0 & 0 & 0.088 & 0.088 & 0.088 & 0.0882 & 0 & 0 & 0.088 \\ 0 & 0 & 0 & 0.043 & 0.043 & 0.043 & 0 & 0 & 0.043 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.467 & 0.467 & 0.004 & 0 & 0.465 & 0.144 & 0.427 & 0.467 & 0.297 & 0.129 \\ 0 & 0 & 0.320 & 0.336 & 0 & 0.336 & 0.336 & 0.336 & 0.336 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

If the number of cluster centers equals $\mathtt{k} = 2$, then the first step of Algorithm 7 consists in choosing the 2 biggest values in the diagonal of s: the first element 0.528 and the 8-th element 0.467 of the diagonal. Then, we round these two values to 1, and the others to 0 to constitute the diagonal of the rounded solution s_f.

$$\begin{bmatrix} 1 & & & & & & & & & \\ & 0 & & & & & & & & \\ & & 0 & & & & & & & \\ & & & 0 & & & & & & \\ & & & & 0 & & & & & \\ & & & & & 0 & & & & \\ & & & & & & 0 & & & \\ & & & & & & & 1 & & \\ & & & & & & & & 0 & \\ & & & & & & & & & 0 \end{bmatrix}.$$

In the next step, we know that the only non zero rows are the first and the 8-th. Thus, we sort every column of the reduced matrix formed with the rows 1 and 8, except for the results based on the diagonal sort that is already done: the first and 8-th column (in blue).

$$s = \begin{bmatrix} 1 & 0 & 0.528 & 0 & 0 & 0 & 0 & 0 & 0 & 0.528 \\ 0 & 0.467 & 0.004 & 0 & 0.465 & 0.144 & 0.427 & 1 & 0.297 & 0.129 \end{bmatrix}.$$

For the rest of the columns, we round to 1 the biggest value, and we round the other to 0, as the following:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

Thus, the rounded solution in its matrix form is the following:

$$s_f = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Note that in Line 3 (resp. Line 4) of Algorithm 7, we need to sort vectors to find the `k` biggest values (resp. the biggest value) of the diagonal (resp. of each column). If there are repetitive values in the sort, then the output s_f is not necessarily unique. In this case, the sorting function chooses randomly the order of these repetitive value. As in DFW Algorithm, we choose the line search step $\gamma^{(\mathring{k})}$ (defined in Algorithm 6, Line 9) because it guarantees that $g(x^{(\mathring{k})})$ decreases at each iteration. The stopping criteria has been chosen as the convergence of the relaxed problem. The following section shows the numerical results that validate the heuristic proposed approach MFW.

IV.5 Numerical results

This section is dedicated to the results of Algorithm 6. First, we describe the experimental setup. Then we compare the obtained solutions and the performance between MFW Algorithm and the exact method using CPLEX for different sizes of the problem.

IV.5.1 Experimental setup

To test the heuristic approach MFW, we consider the robust `k`-median problem with n points, for different values of n. In the numerical experiments, we generate random mean vectors μ and random covariance matrices Σ following the procedure detailed in IV.5.2, and we fix $\Omega = 1$. For the implementation, we use the `Python` language. To compute the solution $s^{(\mathring{k})}$ (Line 8 of Algorithm 6), we used an LP minimizer with the LP modeler `PuLP`. We compare with the BSOCP solver of CPLEX. The random feasible solution $x^{(0)}$ is chosen by first generating a random

cost vector c, and then by solving the k-median problem for this cost vector c (Problem (IV.4)) using the LP minimizer with the LP modeler PuLP to find a feasible solution $x^{(0)}$.

IV.5.2 Adequate μ and Σ generation

To generate adequate μ and Σ, we use Theorem (1): we first generate μ_v and Σ_v randomly, then deduce μ and Σ using Theorem (1): $\mu = L\mu_v$, and $\Sigma = L\Sigma_v L^T$. The matrix L is deduced numerically by a construction of a key matrix that contains the correspondence between the indices in v and d. For the generation of μ_v and Σ_v, the elements of μ_v are chosen randomly in $[0, 100]$. The random covariance matrix Σ_v is defined as in [Ilyina 17]: writing $\Sigma_v = \mathcal{P}^T\mathcal{D}\mathcal{P}$ where $\mathcal{P}$ is an orthogonal eigenvector matrix and $\mathcal{D}$ is the corresponding diagonal eigenvalue matrix, each of the eigenvalues λ_i, $i = 1, \ldots, \frac{n(n-1)}{2}$, is chosen as the square of a random number in $[0, \mu_{v_i}]$ and $\mathcal{P}$ is a random orthogonal matrix.

IV.5.3 Results of MFW for different problem sizes

This part aims at testing the heuristic algorithm MFW by comparing its solutions to the ones obtained by the exact solver of CPLEX for different problem sizes. For this, we change the problem size n, which is the number of points for which a robust k-median problem is to solve. For every problem size n, we choose to find $\mathtt{k} = 2$ cluster centers. The number of iterations for MFW Algorithm is fixed to $\mathring{K} = 200$. For each problem size n, we change the problem 10 times, by changing the random mean vector μ and the random covariance matrix Σ. For each problem, we change x_0 in the input of MFW Algorithm 8 times. Table IV.1 shows for n between 5 and 13 the comparison between MFW Algorithm and the solver of CPLEX. For each problem size n, we compute the mean of the outputs of the different sets (μ, Σ, x_0). These outputs are

1. the processing time (in seconds),
2. the mean of the relative error $\overline{E_r}$ for the different sets of the experimental tests (μ, Σ, x_0), where the relative error is defined as
$$E_r = \frac{g(\hat{x}) - p^*}{p^*},$$
3. and the occurrence of the value zero of the relative error for the different sets of the experimental tests (μ, Σ, x_0)
$$\#\{E_r = 0\}.$$

.

n	Time(s) of CPLEX	Time(s) MFW	$\#\{E_r = 0\}$	$\overline{E_r}$
5	0.1644	5.0149	55 %	0.0555
6	0.5424	7.8	71.25 %	0.0513
7	0.8296	12.9796	48.75 %	0.0486
8	0.9948	6.9707	67.5 %	0.0186
9	1.9202	9.6168	63.75 %	0.0246
10	2.1028	16.6282	58.75 %	0.0432
11	2.1607	14.1045	70 %	0.0447
12	3.6378	19.778	23 %	0.0862
13	4.1873	17.8977	35 %	0.0694

Table IV.1: Comparison of the proposed solution by MFW with the optimal solution by CPLEX.

The fifth column in Table IV.1 shows that the relative error is in average small (0.0186 to 0.0862), and the fourth column shows that the relative error equals zero in up to 70% of the cases. Finally, the two first columns show that CPLEX is faster but the difference in time between MFW Algorithm and CPLEX is not very big (around 1/5 in average).

IV.5.4 Discussion

To sum up the numerical results, MFW Algorithm gives a zero error in many cases, and a small relative error in general. The advantage about this algorithm is that it always gives a feasible solution, due to the feasible rounding algorithm. Many perspectives open up with these experimental results. Mainly, as already mentioned, the sorting algorithm does not give preference to a solution more than another if the rounding is not unique. Thus, an interesting way to easily get smaller relative errors is possible by working on different methods of rounding that give solutions with close-to-optimal objective functions (e.g. see [Charikar 12]). Another fact is that Awasthi *et al.* [Awasthi 15] studied exact recovery conditions for convex relaxations of the `k`-median problem. Thus, in some cases where the clusters are at a great distance from each other, there is no need for a feasible rounding, since the intermediate steps in MFW Algorithm are binary solutions under some mild assumptions over the `k`-median clustering problem detailed in [Awasthi 15]. After working on the perspectives just listed, MFW Algorithm is an algorithm that solves heuristically robust `k`-median clustering problems under uncertainty. This algorithm is adapted for problems with large sizes, where `k`-median clustering is used. For instance, facility problems with large sizes, that are subject to uncertainty with correlated variables (see [Arya 04]).

Conclusions and perspectives

Conclusions

In this manuscript, robust combinatorial optimization has been studied, with a special interest on the ellipsoidal uncertainty set for its multiple advantages.

Since the robust counterpart of optimization problems are harder to solve than the original problems, we worked on the robust counterpart of binary linear problems. In this case, the robust problem is still NP-hard, and can be solved by branch-and-bound methods, that are not adapted for problems with large size. Thus, a Frank-Wolfe based heuristic algorithm named DFW has been proposed in Chapter II for robust binary linear problems with some assumptions that are mainly linked with the exact binarity relaxation. The proposed algorithm is a relaxation-guided version of the Frank-Wolfe algorithm, where we are interested in the optimum of the linear approximation that the algorithm computes at each iteration when relaxing the constraint set in its convex hull. This is legitimate under the assumptions that have been considered, and the shortest path problem is one of the problems that satisfy them. Numerous numerical experiments have been carried for the robust shortest path problem, and comparisons with the optimal solution given by the second order cone programming solver of CPLEX have been done. Results show that DFW approach always gives the same solution as CPLEX in the instances where this solver is able to propose one. In addition, the scalability of our algorithm has been approved in problems of large size, where branch-and-bound methods are no longer efficient.

To avoid comparing the heuristic solution with the optimal one given by the exact method, a validation by lower bound has been studied in Chapter III. This lower bound results from a bidualization of the robust problem. To compute the proposed lower bound, we need to solve an SDP problem that can be solved using interior-point methods. Unfortunately, the bidual problem is a big problem with much more constraints and more variables than the original problem. Thus, despite its polynomial nature, the resolution of this bidual problem is very time consuming and needs a huge memory space. Therefore, the sparsity of the matrices that define the problem has been exploited to replace the classical solver by a sparse version of an algorithm based on the projection of the constraints that is

done by a formalization in a product space. All this is numerically tested, showing that a polynomial time evaluation of the quality of the solution of DFW heuristic is possible without having the memory storage issue of the bidual problem.

Finally, an extension of the algorithm proposed in Chapter II has been studied for the problems that do not satisfy the assumptions of Chapter II. In Chapter IV, the classical k-median problem is written as a binary linear programming problem (BLP) by a matrix flattening step. The relaxation of this problem does not necessarily give binary solutions, and thus a direct application of DFW Algorithm is not possible. Next, the robust k-median clustering problem under ellipsoidal uncertainty is written as a binary non-linear problem. For this, it has been proven in Chapter IV that the matrix flattening step conserves the representation of uncertainty as for the original formulation. For solving the robust k-median problem, the Frank-Wolfe based approach to solve the binary non-linear problem is named MFW Algorithm. It consists first in relaxing the binarity constraints and using the Frank-Wolfe Algorithm for the convex problem. Then, it uses a rounding technique for the mean of the intermediate steps of the algorithm to give a heuristic solution that is a feasible clustering solution. Results show that this approach gives the optimal solution in most of the cases, and that it gives close-to-optimal solutions when they are not optimal.

Perspectives

As perspectives, first, the DFW Algorithm proposed in Chapter II could be improved, to obtain better results and obtain the optimal solution in less processing time, for example using the away step FW [Guélat 86] in order to discover solutions in other directions during the iterations. Next, more numerical results could include a study of the algorithm's parameters, and more tests with larger instances.

Another question to investigate is the possibility to have a proof of sub-optimality. As we know, the proposed approach is heuristic, and thus it has certainly some limitations. Therefore, it would be useful to show that the proposed solution has a certain distance from optimality, in its worst case. It could also be interesting to find out what are the cases where the approach does not give an optimal solution, and what are the cases where it behaves well.

Another direction is the application of DFW Algorithm on a real life application. This has been initiated during several workshops with the society SCODER for a the proposition of a robust metal coils assignment. Here, the possible direction for future work is to measure the gap between the ellipsoidal model of the uncertainty and the real knowledge of the uncertainty in the problem.

Then, a possible future work could be the extension of the heuristic algorithm DFW for more general problems. This has been initiated in Chapter IV and has led to MFW Algorithm, but this perspective needs more investigation.

For the perspectives of Chapter III, several numerical challenges exist, and have been discussed extensively in Section III.2.4 and Section III.2.5. The two most relevant perspectives for this chapter are the following. The first is the acceleration of Pierra's Algorithm that has been proposed with the sparse computations to replace the full version of exact methods. This can be done by parallelization, a parameter's optimization, or by adapting the algorithm's iterations for a better conditioning and a better performance. The second is a comparison with other lower bounds, to compare the optimality gap between the proposed lower bound and other lower bounds.

Finally, Chapter IV has been a first attempt to extend DFW Algorithm for general binary linear problems. It opens to two main perspectives. The first is to work on the proposed algorithm MFW that solves heuristically the robust `k`-median problem, in order to obtain better relative errors. Two possible ways to do so have been described in Section IV.5.4. The second perspective is to apply the heuristic approach MFW on real life examples of `k`-median problems, and observe the obtained relative errors for different uncertainty configurations of the clusters.

Bibliography

[Abernethy 09] J. D. Abernethy, E. Hazan & A. Rakhlin. *Competing in the dark: An efficient algorithm for bandit linear optimization.* 2009.

[Adjiashvili 15] D. Adjiashvili, S. Stiller & R. Zenklusen. *Bulk-robust combinatorial optimization.* Mathematical Programming, vol. 149, no. 1, pages 361–390, 2015.

[Aissi 05] H. Aissi, C. Bazgan & D. Vanderpooten. *Complexity of the min–max and min–max regret assignment problems.* Operations research letters, vol. 33, no. 6, pages 634–640, 2005.

[Aissi 07] H. Aissi, C. Bazgan & D. Vanderpooten. *Approximation of min–max and min–max regret versions of some combinatorial optimization problems.* European Journal of Operational Research, vol. 179, no. 2, pages 281–290, 2007.

[Aissi 09] H. Aissi, C. Bazgan & D. Vanderpooten. *Min–max and min–max regret versions of combinatorial optimization problems: A survey.* European journal of operational research, vol. 197, no. 2, pages 427–438, 2009.

[Al Dahik 20] C. Al Dahik, Z. Al Masry, S. Chrétien, J.-M. Nicod & L. Rabehasaina. *A Frank-Wolfe Based Algorithm for Robust Discrete Optimization under Uncertainty.* In 2020 Prognostics and Health Management Conference (PHM-Besançon), pages 247–252. IEEE, 2020.

[Al Dahik 21] C. Al Dahik, Z. Al Masry, S. Chrétien, J.-M. Nicod & L. Rabehasaina. *An SDP dual relaxation for the Robust Shortest Path Problem with ellipsoidal uncertainty: Pierra's decomposition method and a new primal Frank-Wolfe-type heuristics for duality gap evaluation.* arXiv preprint arXiv:2110.15653, 2021.

[Alizadeh 03] F. Alizadeh & D. Goldfarb. *Second-order cone programming.* Mathematical programming, vol. 95, no. 1, pages 3–51, 2003.

[Anjos 11] M. F. Anjos & J. B. Lasserre. Handbook on semidefinite, conic and polynomial optimization, volume 166. Springer Science & Business Media, 2011.

[Arslan 20] A. Arslan, M. Poss & M. Silva. *Min-max-min robust combinatorial optimization with few recourse solutions*. 2020.

[Arya 04] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala & V. Pandit. *Local search heuristics for k-median and facility location problems*. SIAM Journal on computing, vol. 33, no. 3, pages 544–562, 2004.

[Ausiello 12] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela & M. Protasi. Complexity and approximation: Combinatorial optimization problems and their approximability properties. Springer Science & Business Media, 2012.

[Awasthi 15] P. Awasthi, A. S. Bandeira, M. Charikar, R. Krishnaswamy, S. Villar & R. Ward. *Relax, no need to round: Integrality of clustering formulations*. In Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, pages 191–200. ACM, 2015.

[Baron 19] O. Baron, O. Berman, M. M. Fazel-Zarandi & V. Roshanaei. *Almost Robust Discrete Optimization*. European Journal of Operational Research, vol. 276, no. 2, pages 451–465, 2019.

[Ben-Tal 04] A. Ben-Tal, A. Goryashko, E. Guslitzer & A. Nemirovski. *Adjustable robust solutions of uncertain linear programs*. Mathematical Programming, vol. 99, no. 2, pages 351–376, 2004.

[Ben-Tal 09] A. Ben-Tal, L. El Ghaoui & A. Nemirovski. Robust optimization, volume 28. Princeton University Press, 2009.

[Bertsimas 03] D. Bertsimas & M. Sim. *Robust discrete optimization and network flows*. Mathematical programming, vol. 98, no. 1-3, pages 49–71, 2003.

[Bertsimas 04a] D. Bertsimas, D. Pachamanova & M. Sim. *Robust linear optimization under general norms*. Operations Research Letters, vol. 32, no. 6, pages 510–516, 2004.

[Bertsimas 04b] D. Bertsimas & M. Sim. *Robust discrete optimization under ellipsoidal uncertainty sets*. 2004.

[Bertsimas 19] D. Bertsimas, J. Dunn, C. Pawlowski & Y. D. Zhuo. *Robust classification*. INFORMS Journal on Optimization, vol. 1, no. 1, pages 2–34, 2019.

[Bougeret 19] M. Bougeret, A. A. Pessoa & M. Poss. *Robust scheduling with budgeted uncertainty*. Discrete applied mathematics, vol. 261, pages 93–107, 2019.

[Boyd 94] S. Boyd, L. El Ghaoui, E. Feron & V. Balakrishnan. Linear matrix inequalities in system and control theory, volume 15. Siam, 1994.

[Boyd 11] S. Boyd, N. Parikh & E. Chu. Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc, 2011.

[Buchheim 17] C. Buchheim & J. Kurtz. *Min–max–min robust combinatorial optimization*. Mathematical Programming, vol. 163, no. 1-2, pages 1–23, 2017.

[Buchheim 18a] C. Buchheim, M. De Santis, F. Rinaldi & L. Trieu. *A Frank–Wolfe based branch-and-bound algorithm for mean-risk optimization*. Journal of Global Optimization, vol. 70, no. 3, pages 625–644, 2018.

[Buchheim 18b] C. Buchheim & J. Kurtz. *Robust combinatorial optimization under convex and discrete cost uncertainty*. EURO Journal on Computational Optimization, vol. 6, no. 3, pages 211–238, 2018.

[Buhmann 18] J. M. Buhmann, A. Y. Gronskiy, M. Mihalák, T. Pröger, R. Šrámek & P. Widmayer. *Robust optimization in the presence of uncertainty: A generic approach*. Journal of Computer and System Sciences, vol. 94, pages 135–166, 2018.

[Burgard 15] J. P. Burgard, C. M. Costa & M. Schmidt. *Decomposition Methods for Robustified k-Means Clustering Problems: If Less Conservative Does Not Mean Less Bad*. Rapport technique, Tech. rep. 2020. url: http://www. optimization-online. org/DB_HTML/2020/05 . . . , 2015.

[Charikar 12] M. Charikar & S. Li. *A dependent LP-rounding approach for the k-median problem*. In International Colloquium on Automata, Languages, and Programming, pages 194–205. Springer, 2012.

[Coco 14] A. A. Coco, J. C. A. Júnior, T. F. Noronha & A. C. Santos. *An integer linear programming formulation and heuristics for the minmax relative regret robust shortest path problem.* Journal of Global Optimization, vol. 60, no. 2, pages 265–287, 2014.

[Delage 10] E. Delage & Y. Ye. *Distributionally robust optimization under moment uncertainty with application to data-driven problems.* Operations research, vol. 58, no. 3, pages 595–612, 2010.

[Diamond 16] S. Diamond & S. Boyd. *CVXPY: A Python-embedded modeling language for convex optimization.* The Journal of Machine Learning Research, vol. 17, no. 1, pages 2909–2913, 2016.

[Dijkstra 59] E. W. Dijkstra. *A note on two problems in connexion with graphs.* Numerische mathematik, vol. 1, no. 1, pages 269–271, 1959.

[Fletcher 81] R. Fletcher. *A nonlinear programming problem in statistics (educational testing).* SIAM Journal on Scientific and Statistical Computing, vol. 2, no. 3, pages 257–267, 1981.

[Frank 56] M. Frank & P. Wolfe. *An algorithm for quadratic programming.* Naval research logistics quarterly, vol. 3, no. 1-2, pages 95–110, 1956.

[Gao 11] Y. Gao. *Shortest path problem with uncertain arc lengths.* Computers & Mathematics with Applications, vol. 62, no. 6, pages 2591–2600, 2011.

[Garey] M. R. Garey & D. S. Johnson. Computers and intractability, volume 174.

[Goemans 94] M. X. Goemans & D. P. Williamson. *New 34-approximation algorithms for the maximum satisfiability problem.* SIAM Journal on Discrete Mathematics, vol. 7, no. 4, pages 656–666, 1994.

[Goemans 95] M. X. Goemans & D. P. Williamson. *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming.* Journal of the ACM (JACM), vol. 42, no. 6, pages 1115–1145, 1995.

[Goemans 97] M. X. Goemans. *Semidefinite programming in combinatorial optimization.* Mathematical Programming, vol. 79, no. 1-3, pages 143–161, 1997.

[Goh 10] J. Goh & M. Sim. *Distributionally robust optimization and its tractable approximations.* Operations research, vol. 58, no. 4-part-1, pages 902–917, 2010.

[Guélat 86] J. Guélat & P. Marcotte. *Some comments on Wolfe's 'away step'.* Mathematical Programming, vol. 35, no. 1, pages 110–119, 1986.

[Gut] A. Gut. *The Multivariate Normal Distribution.* In An Intermediate Course in Probability, pages 117–145. Springer New York.

[Haddad 21] M. Haddad, G. Da Costa, J.-M. Nicod, M.-C. Péra, J.-M. Pierson, V. Rehn-Sonigo, P. Stolf & C. Varnier. *Combined IT and power supply infrastructure sizing for standalone green data centers.* Sustainable Computing: Informatics and Systems, vol. 30, page 100505, 2021.

[Hanasusanto 15] G. A. Hanasusanto, D. Kuhn & W. Wiesemann. *K-adaptability in two-stage robust binary programming.* Operations Research, vol. 63, no. 4, pages 877–891, 2015.

[Hanasusanto 16] G. A. Hanasusanto, D. Kuhn & W. Wiesemann. *K-adaptability in two-stage distributionally robust binary programming.* Operations Research Letters, vol. 44, no. 1, pages 6–11, 2016.

[Hazan 16] E. Hazan*et al.* *Introduction to online convex optimization.* Foundations and Trends® in Optimization, vol. 2, no. 3-4, pages 157–325, 2016.

[Ilyina 17] A. Ilyina. *Combinatorial optimization under ellipsoidal uncertainty.* PhD thesis, Technische Universität Dortmund, 2017.

[Jaggi 13] M. Jaggi. *Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization.* In ICML (1), pages 427–435, 2013.

[Kacem 19] I. Kacem & H. Kellerer. *Complexity results for common due date scheduling problems with interval data and minmax regret criterion.* Discrete Applied Mathematics, vol. 264, pages 76–89, 2019.

[Karger 98] D. Karger, R. Motwani & M. Sudan. *Approximate graph coloring by semidefinite programming.* Journal of the ACM (JACM), vol. 45, no. 2, pages 246–265, 1998.

[Karloff 99] H. Karloff. *How Good is the Goemans–Williamson MAX CUT Algorithm?* SIAM Journal on Computing, vol. 29, no. 1, pages 336–350, 1999.

[Kouvelis] P. Kouvelis & G. Yu. Robust discrete optimization and its applications. OCLC: 854966265.

[Lee 04] J. Lee. A first course in combinatorial optimization, volume 36. Cambridge University Press, 2004.

[Lemaréchal 99] C. Lemaréchal & F. Oustry. *Semidefinite relaxations and Lagrangian duality with application to combinatorial optimization.* 1999.

[Li 11] Z. Li, R. Ding & C. A. Floudas. *A comparative theoretical and computational study on robust counterpart optimization: I. Robust linear optimization and robust mixed integer linear optimization.* Industrial & engineering chemistry research, vol. 50, no. 18, pages 10567–10603, 2011.

[Li 16] J. Li, S. Song, Y. Zhang & Z. Zhou. *Robust k-median and k-means clustering algorithms for incomplete data.* Mathematical Problems in Engineering, vol. 2016, 2016.

[Liebchen 09] C. Liebchen, M. Lübbecke, R. Möhring & S. Stiller. *The concept of recoverable robustness, linear programming recovery, and railway applications.* In Robust and online large-scale optimization, pages 1–27. Springer, 2009.

[Liu 09] B. Liu. *Some research problems in uncertainty theory.* Journal of Uncertain systems, vol. 3, no. 1, pages 3–10, 2009.

[Manual 87] C. U. Manual. *Ibm ilog cplex optimization studio.* Version, vol. 12, pages 1987–2018, 1987.

[Markowitz 52] H. Markowitz. *Portfolio selection.* The journal of finance, vol. 7, no. 1, pages 77–91, 1952.

[Nesterov 94] Y. Nesterov & A. Nemirovski. *Interior-Point Polynomial Algorithms in Convex Programming (Studies in Applied and Numerical Mathematics).* Society for Industrial Mathematics, 1994.

[Nocedal 06] J. Nocedal & S. Wright. Numerical optimization. Springer Science & Business Media, 2006.

[Omri 20] N. Omri, Z. Al Masry, N. Mairot, S. Giampiccolo & N. Zerhouni. *Industrial data management strategy towards an SME-oriented PHM*. Journal of Manufacturing Systems, vol. 56, pages 23–36, 2020.

[Pant 11] R. Pant, T. B. Trafalis & K. Barker. *Support vector machine classification of uncertain and imbalanced data using robust optimization*. In Proceedings of the 15th WSEAS international conference on computers, pages 369–374. World Scientific and Engineering Academy and Society (WSEAS) Stevens Point . . . , 2011.

[Pierra 84] G. Pierra. *Decomposition through formalization in a product space*. Mathematical Programming, vol. 28, no. 1, pages 96–115, 1984.

[Pierson 19] J.-M. Pierson, G. Baudic, S. Caux, B. Celik, G. Da Costa, L. Grange, M. Haddad, J. Lecuivre, J.-M. Nicod, L. Philippe*et al.* *Datazero: Datacenter with zero emission and robust management using renewable energy*. IEEE Access, vol. 7, pages 103209–103230, 2019.

[Portoleau 20] T. Portoleau, C. Artigues & R. Guillaume. *Robust Predictive-Reactive Scheduling: an Information-Based Decision Tree Model*. In International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, pages 479–492. Springer, 2020.

[Poss 13] M. Poss. *Robust combinatorial optimization with variable budgeted uncertainty*. 4OR, vol. 11, no. 1, pages 75–92, 2013.

[Poss 18] M. Poss. *Robust combinatorial optimization with knapsack uncertainty*. Discrete Optimization, vol. 27, pages 88–102, 2018.

[Rahimian 19] H. Rahimian & S. Mehrotra. *Distributionally robust optimization: A review*. arXiv preprint arXiv:1908.05659, 2019.

[Reeves 93] C. R. Reeves, editeur. Modern heuristic techniques for combinatorial problems. John Wiley & Sons, Inc., USA, 1993.

[Saltelli 02] A. Saltelli. *Sensitivity analysis for importance assessment*. Risk analysis, vol. 22, no. 3, pages 579–590, 2002.

[Scobey 78] P. Scobey & D. Kabe. *Vector quadratic programming problems and inequality constrained least squares estimation*. J. Indust. Math. Soc., vol. 28, pages 37–49, 1978.

[Shalev-Shwartz 07] S. Shalev-Shwartz & Y. Singer. *A primal-dual perspective of online learning algorithms*. Machine Learning, vol. 69, no. 2, pages 115–142, 2007.

[Singla 20] M. Singla, D. Ghosh & K. Shukla. *A survey of robust optimization based machine learning with special reference to support vector machines*. International Journal of Machine Learning and Cybernetics, vol. 11, no. 7, pages 1359–1385, 2020.

[Soyster 73] A. L. Soyster. *Convex programming with set-inclusive constraints and applications to inexact linear programming*. Operations research, vol. 21, no. 5, pages 1154–1157, 1973.

[Toklu 17] N. E. Toklu, S. Yanık & R. Montemanni. *THE TRAVELING SALESMAN PROBLEM UNDER DYNAMIC UNCERTAINTY*. 2017.

[Trafalis 07] T. B. Trafalis & R. C. Gilbert. *Robust support vector machines for classification and computational issues*. Optimisation Methods and Software, vol. 22, no. 1, pages 187–198, 2007.

[Wolkowicz 99] H. Wolkowicz. *Semidefinite and Lagrangian relaxations for hard combinatorial problems*. In IFIP Conference on System Modeling and Optimization, pages 269–309. Springer, 1999.

[Xanthopoulos 12] P. Xanthopoulos, P. M. Pardalos & T. B. Trafalis. Robust data mining. Springer Science & Business Media, 2012.

[Zinkevich 03] M. Zinkevich. *Online convex programming and generalized infinitesimal gradient ascent*. In Proceedings of the 20th international conference on machine learning (icml-03), pages 928–936, 2003.

SPIM

Titre : Optimisation Discrète Robuste En Présence D'incertitude Ellipsoïdale

Mots clefs: Incertitude, Optimisation robuste discrète, Ensemble d'incertitude, Plus court chemin robuste, Clustering robuste, relaxation SDP

Résumé : Cette thèse traite la version robuste des problèmes linéaires à variables binaires avec un ensemble d'incertitude corrélé. Puisque ce problème est NP-difficile, une approche heuristique intitulée DFW et basée sur l'algorithme de Frank-Wolfe est proposée. Dans cette approche, nous examinons la puissance d'exploration des itérations internes binaires de la méthode. Pour les problèmes de petites tailles, la méthode est capable de fournir la solution optimale fournie par CPLEX, après quelques centaines d'itérations. De plus, contrairement à la méthode exacte, notre approche s'applique à des problèmes de grandes tailles également. Les résultats numériques ont été appliqués au plus court chemin robuste. Un autre objectif de cette thèse est de proposer une relaxation semi-définie positive (SDP) pour le plus court chemin robuste qui fournit une borne inférieure pour valider des approches telles que l'algorithme DFW. Le problème relaxé est le résultant d'une bidualisation du problème. Puis le problème relaxé est résolu en utilisant une version creuse d'une méthode de décomposition dans un espace produit. Cette méthode de validation est adaptée aux problèmes de grande taille. Finalement, une autre adaptation de l'algorithme de Frank-Wolfe a été réalisé pour le problème du `k`-médiane, accompagnée d'un algorithme d'arrondissement qui satisfait les contraintes.

Title : Robust Discrete Optimization Under Ellipsoidal Uncertainty

Keywords : Uncertainty, Robust discrete optimization, Ellipsoidal uncertainty set, Robust shortest path Problem, Robust clustering, SDP relaxation

Abstract : This thesis addresses the Robust counterpart of binary linear problems with ellipsoidal uncertainty sets. Since this problem is hard, a heuristic approach, based on Frank- Wolfe's algorithm named Discrete Frank-Wolf (DFW), has been proposed. In this approach, we make use of the exploration power of the integer inner iterates of the method. For small dimensional instances, our method is able to provide the same optimal integer solution as an exact method provided by CPLEX, after no more than a few hundred iterations. Moreover, as opposed to the exact method, DFW Algorithm applies to large scale problems as well. The numerical results are applied on the robust shortest path problem (RSPP). Another aim of this thesis is to propose a Semi-Definite Programming (SDP) relaxation for the RSPP that provides a lower bound to validate approaches such as DFW Algorithm. The relaxed problem results from a bidualization of the problem.Then the relaxed problem is solved using a sparse version of a decomposition in a product space method. This validation method is suitable for large size problems. The numerical experiments show that the gap between the solutions obtained with the relaxed and the heuristic approaches is relatively small. Finally, another adaptation of FW, named MFW Algorithm, has been proposed and tested numerically for the `k`-median problem with a feasible rounding procedure.

UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besançon