APPLICATIONS AND ALGORITHMS FOR TWO-STAGE ROBUST LINEAR OPTIMIZATION

Marco Aurelio Costa da Silva

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Nelson Maculan Filho
Michaël Jérémie Poss

Rio de Janeiro
Novembro de 2018

APPLICATIONS AND ALGORITHMS FOR TWO-STAGE ROBUST LINEAR OPTIMIZATION

Marco Aurelio Costa da Silva

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Nelson Maculan Filho, D.Sc.

Prof. Michaël Jérémie Poss, D.Sc.

Prof. Abilio Pereira de Lucena Filho, D.Sc.

Prof. Laura Silvia Bahiense da Silva Leite, D.Sc.

Prof. Claudia Alejandra Sagastizábal, D.Sc.

Prof. Mario Veiga Ferraz Pereira, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
NOVEMBRO DE 2018

Silva, Marco Aurelio Costa da

Applications and algorithms for two-stage robust linear optimization/Marco Aurelio Costa da Silva. – Rio de Janeiro: UFRJ/COPPE, 2018.

XII, 116 p.: il.; 29, 7cm.

Orientadores: Nelson Maculan Filho

Michaël Jérémie Poss

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p. 105 – 116.

1. Robust linear optimization. 2. Conservatism. 3. min-max-min robust optimization. 4. Uncertainty sets. 5. Data-driven optimization. 6. Distributionally robust optimization. 7. Network design. 8. Scheduling. 9. Fleet management. I. Maculan Filho, Nelson *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

to my wife Sheila, for everything
to my son Alan, and my
daughter Thais, for being such a
joy in my life
for my parents, Luiz and
Cristina, for being such a great
reference

Acknowledgements

At first, I would like to thank my supervisors, Nelson Maculan and Michael Poss. I could not have imagined better sources of wisdom and knowledge.

I am most grateful to Michael, who introduced me to robust optimization with a contagious enthusiasm. His patience, guidance and support were of inestimable value. I also thank him for a friendship that made my stay at Avignon so remarkable.

Professor Maculan is an inspiring leader. I have witness how he has changed the life of so many people for the best and he also made a big difference in my life. I would not have started this journey at COPPE/PESC if it wasn't for his will to trust me.

I thank professor Philippe Michelon for giving me a precious opportunity at the University of Avignon where I was able to develop my knowledge to a level it would not be possible elsewhere.

Professor Marcia Fampa was always a reference to me of what it is to be a real researcher. Her words of wisdom and her shared experience have made my life so much easier and so much fun. I am so grateful for her companionship.

All this would not have been possible also if it wasn't for Professor Rosa Figueiredo. I cherish her friendship and have not enough words to thank all the support she has given me and my family these years. I have her as a reference for my future.

I am also grateful to Professor Francois Vanderbeck and to Issam Tahiri who have introduced me to BapCod and column generation and have made my stay at University of Bordeaux so much profitable.

I am honored that Mr. Adam Ouorou and Professor Dritan Nace have accepted the role of *rapporteurs* for my thesis and that Professor Abilio Lucena, Professor Laura Bahiense, Professor Claudia Sagastizabal and Mr Mario Veiga have accepted to compose the thesis jury.

Finally, I have enjoyed the company of some great colleagues and friends over these years. Mohamed Bouaziz, Etienne Papegnies, Bruno Rosa, Thiago Gouveia, Elvys Pontes, Marcio Santos and Luiz Flores always provided me with different perspectives.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

APLICAÇÕES E ALGORITMOS PARA OTIMIZAÇÃO LINEAR ROBUSTA EM DOIS ESTÁGIOS

Marco Aurelio Costa da Silva

Novembro/2018

Orientadores: Nelson Maculan Filho
Michaël Jérémie Poss

Programa: Engenharia de Sistemas e Computação

O âmbito de pesquisa desta tese é otimização linear robusta em dois estágios. Estamos interessados em investigar algoritmos que exploram a sua estrutura e também em alternativas que se somem para mitigar o conservadorismo inerente da otimização robusta.

Nós desenvolvemos algoritmos que incorporam estas alternativas e que são orientados para instâncias de problemas de média e larga escala.

Fazendo isto experimentamos uma abordagem holística para analisar o conservadorismo em otimização linear robusta e integramos os mais recentes avanços em áreas como a otimização robusta baseada em dados históricos (*data-driven robust optimization*), otimização robusta distribucional (*distributionally robust optimization*) e otimização robusta ajustável (*adaptive robust optimization*).

Nós exercitamos estes algoritmos em aplicações definidas de problemas de projeto de redes (*network design/loading*), escalonamento (*scheduling*), *min-max-min* combinatoriais particulares e atribuição de frotas na aviação (*airline fleet assignment*); e mostramos como os algoritmos desenvolvidos melhoram performance quando comparados com implementações anteriores.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

APPLICATIONS AND ALGORITHMS FOR TWO-STAGE ROBUST LINEAR OPTIMIZATION

Marco Aurelio Costa da Silva

November/2018

Advisors: Nelson Maculan Filho
Michaël Jérémie Poss

Department: Systems Engineering and Computer Science

The research scope of this thesis is two-stage robust linear optimization. We are interested in investigating algorithms that can explore its structure and also on adding alternatives to mitigate conservatism inherent to a robust solution.

We develop algorithms that incorporate these alternatives and are customized to work with rather medium or large scale instances of problems.

By doing this we experiment a holistic approach to conservatism in robust linear optimization and bring together the most recent advances in areas such as data-driven robust optimization, distributionally robust optimization and adaptive robust optimization.

We apply these algorithms in defined applications of the network design/loading problem, the scheduling problem, a min-max-min combinatorial problem and the airline fleet assignment problem. We show how the algorithms developed improve performance when compared to previous implementations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The overall theme of this thesis is optimization under data uncertainty with a robust optimization approach. Robust optimization, as introduced by [17] and [65], assumes a non stochastic nature of uncertainty or that the probability distribution of the uncertainty data is unknown. It models the uncertainty data as belonging to a so called uncertainty set. Within robust optimization a feasible solution is defined as a solution that will remain feasible for all possible realizations of data and the objective is defined by the worst-case performance.

Robust optimization has emerged as a successful discipline because of the general tractability of its formulations, specially when compared to stochastic optimization, another approach to deal with data uncertainty that assumes a stochastic nature of uncertainty and known or partially known probability distributions.

On the other hand, the pessimist worst-case approach of robust optimization yields to solutions that can be too conservative in practice, meaning that it will perform best if worst-case scenario happens, but perform bad otherwise. This conservatism has been characterized under the expression "price of robustness" in the seminal work in [31], in the sense that there is a trade off between feasibility and optimality.

A well known framework to mitigate this conservatism is two-stage robust optimization. In this thesis we focus on formulations that are two-stage robust and represent a robust mixed-integer linear optimization problem (MILP). Two-stage robust optimization will be detailed as part of a larger context of different alternatives to mitigate the conservatism in Chapter 2.

We investigate how these different alternatives work within a framework of two-stage robust optimization, while verifying the implications of these alternatives in terms of formulations and algorithmic approaches. In fact, although robust optimization emerged as a worst-case deterministic alternative to stochastic optimization, since the beginning a lot of the research has focused on connecting these paradigms, so that the models can best leverage the available information.

A detailed list of our objectives for this thesis is:

- Verify what assumptions of the robust paradigm can be changed to add flexibility to its approach while maintaining its tractability attractiveness.
- Develop algorithms for two-stage robust MILP formulations that additionally incorporate these changed assumptions and are customized to work with rather medium or large scale instances of problems.
- Apply these algorithms in defined applications.

Our contributions are in the sense that we improve existing algorithms to solve close to realistic models and they are further detailed in the following outline of our work.

In Chapter 2 we present a theory overview and introduce the main concepts that will be used throughout the thesis. We investigate three streams of work in this area and their implication to conservatism mitigation and complexity of associated resolution algorithms. Namely, (1) we review the definition of different geometries of the uncertainty set and study data-driven uncertainty sets, (2) we verify, under the framework of adjustable or adaptative robust optimization, the use of recourse variables that are defined only after uncertainty is revealed and (3) we review the introduction of different formulations for the objective function, with a extended focus to distributionally robust optimization where it is assumed that uncertainty probability distribution is partially known.

Chapter 3 exercises a robust network design/loading application using a recourse model defined as finite adaptability. We see that finite adaptability is equivalent to the partition of the uncertainty set where a recourse variable is associated with each element of the partition. We use a method that improves this partitioning iteratively, but that on the other hand augments the number of variables substantially at each iteration. We then develop decomposition schemes and valid inequalities to improve performance of the associated resolution algorithms. Historically, partitioning schemes for the uncertainty set had already been proposed for the network design problem, but we show that our finite adaptability algorithm outperforms previous approaches.

Chapter 4 is about a robust scheduling problem where we develop a branch-and-bound algorithm based on dominance rules and a lower bound solved with the aid of dynamic programming or defined heuristics. We leverage the properties of the budget uncertainty set as defined in [31]. We also develop mixed integer formulations with dominance rules for the problem and show in what conditions our branch-and-bound algorithm outperforms the more traditional mixed integer problem.

Chapter 5 treats a robust combinatorial optimization problem where all variables are recourse variables and the uncertainty set is a polyhedral uncertainty set. We

develop a branch-and-price algorithm based on new formulations and on previous intelligent enumeration of feasible solutions.

In Chapter 6 we introduce demand uncertainty to the airline fleet assignment problem. We explore the generation of data-driven uncertainty sets to leverage the existence of historical data and introduce a distributionally robust optimization formulation to reformulate the objective function due to the nature of the problem. Here we benchmark against current deterministic and robust fleet assignment formulations and verify solution performance results through simulation.

Each of the chapters above is self-contained and can be read in any order, although a previous reading of Chapter 2 is recommended for an introductory systemic view of the subject under study. The chapters replicate the necessary theory concepts, when needed.

Chapters 3 and 4 present complete works which have led to publications (Chapter 4 is currently under review). In contrast, Chapters 5 and 6 present preliminary results. Nevertheless, we are confident that these two chapters will soon be extended and submitted to appropriate journals.

Finally Chapter 7 consolidates the main results achieved with this work and delineates future directions.

Chapter 2

Theory review

2.1 Robust optimization

An uncertain linear optimization problem is a collection of linear optimization problem instances, $\min_x\{c^T x : Ax \leq b\}$, of common structure given by m constraints and n variables, $x \in \mathbb{R}^n$, and with varying uncertain data (c, A, b). This collection of optimizations problems is not associated by itself with the concepts of feasibility and optimality. The definition of these concepts will depend on the approach defined to treat uncertainty.

The classical robust approach here defined has the following characteristics:

- Data (c, A, b) varies within a known uncertainty set U. It is assumed that the uncertainty set is parameterized, in a affine fashion, by a perturbation vector ξ that in its turn varies within a known uncertainty set. We can normalize sets with simple geometry (parallelotopes, ellipsoids and others) to standard sets (unit box, unit ball and others).

- All decision variables represent "here and now" decisions, meaning that they are defined before any realization of uncertainty is known.

- The constraints are "hard", meaning that violations of constraints are not tolerated when the data is within the underspecified uncertainty set U.

This way, a vector $x \in \mathbb{R}^n$ is a feasible solution, called robust feasible solution, if it satisfies all realizations of the constraints within the uncertainty set, that is, $Ax \leq b \quad \forall\, (c, A, b) \in U$.

Additionally, following these worst-case oriented assumptions, and given a candidate solution $x \in \mathbb{R}^n$, the robust value of the objective function at x is the worst (largest) value of the objective function over all realizations of the data in the uncertainty set.

We can then define a deterministic optimization problem, called the robust counterpart, to formulate our robust problem:

$$\min_{x} \left\{ \max_{c \in U_c} c^T x : Ax \leq b, \forall\, (A,b) \in U_{(A,b)} \right\}, \tag{2.1}$$

where $U_c = \{c \mid \exists (c, A, b) \in U\}$ and $U_{(A,b)} = \{(A, b) \mid \exists (c, A, b) \in U\}$ denote the projections of U on the coordinates corresponding to c and (A, b), respectively.

Moreover, problem (2.1) can be reformulated equivalently to problem (2.3) if we consider uncertainty restricted to the coefficients of matrix A. This is done by defining a new matrix $A^{'}$ as

$$A^{'} = \begin{bmatrix} A & \mathbf{0} & -b \\ c^T & -1 & 0 \\ \mathbf{0}^T & 0 & 1 \\ \mathbf{0}^T & 0 & -1 \end{bmatrix} \tag{2.2}$$

and defining a new set of variables $x^{'} = (x, y, z)$, a new vector $c^{'} = (0, 1, 0)$ and a new vector $b^{'} = (\mathbf{0}, 0, 1, -1)$ such that

$$\min_{x^{'}} \left\{ c^{'T} x^{'} : A^{'} x^{'} \leq b^{'}, \forall\, A^{'} \in U \right\}, \tag{2.3}$$

where $\mathbf{0}$ are zero vectors of appropriate dimension.

Note also that the robust counterpart is a constraint-wise construction, meaning that each constraint can be given as

$$a_i^T x \leq b_i, \forall\, (a_i, b_i) \in U_i \tag{2.4}$$

where a_i^T is the i-th row in A and U_i is the projection of U on the space of data of the i-th constraint: $U_i = \{(a_i, b_i) \mid (A, b) \in U\}$.

These definitions have an important consequence: if we model the robust counterpart as a linear problem with a certain objective function, it remains intact if we extend the uncertain set U as the direct product of the convex hull of the projections of U on the space of data of each constraint, namely, $conv(U_1) \times \ldots \times conv(U_m)$ (see [20, pp. 11–13] for details and a formal proof).

This potentially adds conservatism to the problem, so that special care should be taken on how the uncertainty set is modeled and on relations between uncertain data that are not expressed within the space of data of each constraint.

In fact, a pioneer work in [114] considered a special case where each column vector of the constraint matrix A is only known to belong to a convex set, a case of column-wise uncertainty. The relations between uncertain data are only defined

between different constraints. This way, the projection of the uncertainty set over each constraint will give rise to interval sets, where each uncertain data of the constraint will vary between each its lower and upper bounds.

Called a box uncertainty set and deemed too conservative for practical implementation, especially when extremes values of uncertainty occurs rarely (associating a stochastic nature to uncertainty). On the other hand, a reformulation of this model leads to also a linear problem that is easy to solve.

2.2 Uncertainty sets

The modeling of the uncertainty sets as a way to reduce conservatism while maintaining tractability was object of the research that followed. In this subsection we consider uncertainty restricted to the coefficients of matrix A.

In [18] and [17] the authors introduced row-wise ellipsoidal uncertainty sets that turned linear programming problems into second-order cone problems and reduced the conservatism of the approach in [114]. Besides the tractability of its robust counterpart, they introduce ellipsoidal uncertainty sets for the following reasons.

- Ellipsoidal uncertainty sets form a relatively wide family including polytopes which can be modeled as the intersection of degenerated ellipsoids.

- An ellipsoid is given parametrically by data of moderate size, hence it is convenient to represent "ellipsoidal uncertainty" as input.

- Although no underlying stochastic model of the data is assumed to be known or even to exist, in many cases of stochastic uncertain data there are probabilistic arguments allowing to replace stochastic uncertainty by an ellipsoidal deterministic uncertainty. Consider an uncertain linear programming problem with random entries in the constraint matrix. For a given x, the left-hand-side $l_i(x) = a_i^T x - b_i$ of the i-th constraint in the system $Ax - b \leq 0$ is a random variable with expectation $\mathbb{E}_i[x] = (a_i^*)^T x + b_i$, and standard deviation $\mathbb{S}_i[x] = (x^T \Sigma_i x)^{\frac{1}{2}}$, a_i^* being the expectation and Σ_i being the covariance matrix of the random vector a_i. For a "light tail" distribution of the random data a "likely" upper bound on this random variable is $\hat{l}_i(x) = \mathbb{E}_i[x] + \theta \mathbb{S}_i[x]$ with "safety parameter" θ of order of one. This bound leads to the "likely reliable" version $\mathbb{E}_i[x] + \theta \mathbb{S}_i[x] \leq 0$ of the constraint in question. It can be shown that the latter constraint is exactly the robust counterpart $a_i^T x \leq b_i, \forall\, a_i \in U_i$ of the original uncertain constraint if U_i is specified as the ellipsoid $U_i = \{a : (a - a_i^*)^T \Sigma_i^{-1} (a - a_i^*) \leq \theta^2\}$. This is because

the robust counterpart can be equivalently defined as $\sup_{a_i \in U_i} (a_i^T x) \leq b_i$ and this will lead equivalently to $\mathbb{E}_i[x] + \theta \mathbb{S}_i[x] \leq 0$ (see [20, pp. 18–19]).

The authors in [20] introduced uncertainty of stochastic nature to robust optimization. By doing this they developed methods to produce less conservative solutions with certain probabilistic guarantees of feasibility. Here, an uncertainty set can be constructed for random parameters a_i whose distribution $\mathbb{P}$, is not known except for some structural features. Furthermore, instead of insisting the given constraint hold almost surely with respect to $\mathbb{P}$, they instead authorize a small probability of violation. Specifically, given $\epsilon > 0$, these approaches seek sets U_ϵ that implies a probabilistic guarantee for $\mathbb{P}$ at level ϵ. Since $\mathbb{P}$ is not known exactly we have the implication:

$$\text{If } a_i^T x - b_i \leq 0, \forall\, a_i \in U_\epsilon, \text{ then } \sup_{\mathbb{P} \in \mathcal{D}} \mathbb{P}(a_i^T x - b_i \leq 0) \geq 1 - \epsilon \tag{2.5}$$

where $\mathcal{D}$ is a family of probability measures known to contain $\mathbb{P}$. The expression $a_i^T x - b_i \leq 0, \forall\, a_i \in U_\epsilon$ is called a safe convex approximation of the worst-case chance constraint $\sup_{\mathbb{P} \in \mathcal{D}} \mathbb{P}(a_i^T x - b_i \leq 0) \geq 1 - \epsilon$.

In fact, since worst-case chance constraints as described in (2.5) can be nonconvex, many authors develop safe convex approximations S as conservative approximations of these constraints. A safe convex approximation S is a system of convex constraints on x and additional variables v such that the x component of every feasible solution (x, v) of S is feasible for the chance constraint. By replacing the chance constraints in a given chance constrained optimization problem with their safe convex approximations, we end up with a convex approximation problem in x and other variables. The idea is to arrive to safe convex approximations that are computationally tractable.

To that end, the authors in [31] investigated the special case where the uncertainty set is a polyhedron. Here each uncertain element of the constraint matrix A is modeled as an independent bounded random variable. They assumed that the probability distributions of the uncertain elements are unknown except that they are symmetric. They used a set of parameters that "restrict" the variability of the uncertain elements. This approach was a breakthrough because it preserves the degree of complexity of the problem (the robust counterpart of a linear problem is linear). The difference between the objective values of the nominal formulation and their robust formulation was termed the price of robustness. For a probabilistic guarantee of feasibility of an optimal solution of their robust formulation, they established upper bounds on the constraint violation probability of the solution.

Further studies on polyhedral uncertainty sets have identified other opportunities

to reduce conservatism. In [39] they verify that the single deviation band for each coefficient proposed in [31] may be too limitative in practice, so that getting a higher resolution by partitioning the band into multiple sub-bands seems advisable. They define for each uncertainty coefficient a partitioning into sub bands and, for each sub band, maximum and minimum values of the parameter under uncertainty.

In [97], the author introduces the concept of decision dependent uncertainty as an extension of the work presented in [31]. He identifies that the uncertainty set proposed in [31] suffers from a practical drawback since they are independent from the value of decision variables, so that some decision vectors, for instance in combinatorial problems with few non-zero components, are more protected than others. He proposes a new model where the uncertain parameters belong to the image of multi-functions of the problem decision variables. It can provide the same probabilistic guarantee as the uncertainty set defined in [31] while being less conservative. The feasibility set of the resulting optimization problem is in general non-convex so that a mixed-integer programming reformulation for the problem is proposed for uncertainty sets with affine dependency on decision variables.

Other authors explored safe convex approximations to develop new methods to create less conservative uncertainty sets, leveraging the existence of historical random data.

A prominent and practical issue of uncertainty set-induced robust optimization is how to determine the set coefficients appropriately, which are in general assumed as known according to domain-specific knowledge. In the absence of first-principle knowledge, leveraging available historical data provides a practical way to characterize the distributional information. In fact the true information available to the decision maker is historical data, which must be incorporated into an uncertainty set before the robust optimization approach can be implemented.

In [25] the authors utilize coherent risk measures μ (see [7] for definition), as safe convex approximations to chance constraints. The measure μ captures the attitude toward risk of the decision maker. Their approach is data-driven, in the sense that they utilize historical data of the random vector a_i. The only information available is a sample set of size N. They assume that a_i is a discrete random variable with support given by $\mathcal{S} = \{a_i^1, \ldots, a_i^N\}$. The authors emphasize that, although this assumption admittedly has its shortcomings, it can be useful in applications where there are no other information on the distribution of a_i. They use the fact that any coherent risk measure can be represented as the worst-case expected value over a family of measures $\mathcal{Q}$, where:

$$\mu(a_i^T x) = \sup_{\mathbb{Q} \in \mathcal{Q}} \mathbb{E}_{\mathbb{Q}}[-a_i^T x]$$

where for all $\mathbb{Q} \in \mathcal{Q}$, $\quad \mathbb{Q}(a_i^T x) = 0$ s.t. $\mathbb{P}(a_i^T x) = 0$.

With these assumptions over a_i and if μ is coherent they prove that:

$$\{x \in \mathbb{R}^n : \mu(a_i^T x) \geq b_i\} = \{x \in \mathbb{R}^n : a_i^T x \geq b_i, \forall\, a_i \in U\},$$

where

$$U = conv\left(\left\{\sum_{j=1}^{N} q_j a_i^j : q \in \mathcal{Q}\right\}\right),$$

This provides a methodology for constructing uncertainty sets U. The uncertainty set is convex and its structure depends on the generating family $\mathcal{Q}$ for μ and the data set $\mathcal{S}$. They explore a subclass of coherent risk measures and show they are equivalent to polyhedral uncertainty sets of a special structure. Note that the convex uncertainty set U is included in the convex hull of the data points $a_i^1, \ldots, a_i^N$.

This work is further generalized in [67], where the author offers a study of connection between hypothesis testing and uncertainty set design. The author suggests the following general schema to construct an uncertainty set:

- Calculate $P(\mathcal{S}, \alpha)$, a confidence region of the probability measures based on a hypothesis testing at test level α.
- Construct a safe approximation function $\phi(x, \mathcal{S}, \alpha, \epsilon)$, such that $\phi(x, \mathcal{S}, \alpha, \epsilon) \geq \sup_{\mathbb{P} \in P(\mathcal{S})} VAR_\epsilon^{\mathbb{P}}(a_i^T x)$, $\forall\, x \in R^n$ and $VAR_\epsilon^{\mathbb{P}}(a_i^T x)$ is the Value at Risk at level ϵ with respect to $a_i^T x$, that is, $VAR_\epsilon^{\mathbb{P}}(a_i^T x) = \inf\{t \mid \mathbb{P}(a_i^T x \leq t) \geq 1 - \epsilon\}$.
- Identify the uncertainty set $U(\mathcal{S}, \alpha, \epsilon)$ whose support function coincides with $\phi(x, \mathcal{S}, \alpha, \epsilon)$.

Their schema depends on a-priori assumptions of $\mathbb{P}$, historical data and a hypothesis test. By pairing different a-priori assumptions and hypothesis test they obtain distinct data-driven uncertainty sets, implying probabilities guarantees.

In [21] the authors introduce data-driven uncertainty sets where the probability measure parameters are uncertain and consider ϕ-divergence distances as a way to construct robust solutions that are feasible for all allowable distributions of the uncertain parameters with bounded support.

In [131] the authors use ϕ-divergence goodness-of-fit statistics for general parameters uncertainty.

Exploring mainly the existence of massive historical data, another approach to create uncertainty sets has been by the use of machine learning techniques. Learning models can be used to provide representations of data distributions. In [108] the

authors view the construction of the uncertainty set as an unsupervised learning problem. They propose the support vector clustering method as a way to estimate the support of an unknown probability distribution from random data samples. Additionally, they develop a novel piecewise linear kernel that leads to a convex polyhedral uncertainty set, thereby rendering the robust counterpart problem of the same type as the deterministic problem.

In [89] the authors argue that support vector clustering techniques suffer from the curse of dimensionality and propose another schema where the support of data distributions are calculated based on a mix of principal component analysis techniques, that are use to identify the latent uncorrelated uncertainties behind observed uncertainty data, and kernel density estimation methods, that are used to capture the probability distribution of the uncertain data projected onto each principal component. They also develop polyhedral uncertainty sets.

2.3 Adaptive robust optimization

As already seen, classical robust optimization problems are modeled under the assumption that all variables are "here-and-now" decision variables. This means that they are defined before any realization of the uncertainty set, and therefore lead to conservative solutions. In many cases, however, decision variables can be defined only after realization of uncertainty. This is the case, for instance, when decisions are made subsequently. Defining a set of variables as "wait-and-see" variables, where they are defined only after all or part of the uncertainty is revealed, can lead to less conservative solutions. The reduction in conservatism is because it yields more flexible decisions that can be adjusted according to the realized portion of the uncertain data.

A decision variable that can incorporate information already revealed by uncertainty is also called adaptive or adjustable decision variable and the actual function that maps the revealed information to the action that is implemented is referred to as a decision rule. The resulting problem is named adaptive or adjustable robust optimization (ARO) problem and the concept was first introduced to robust optimization in [19]. Adaptive robust optimization has many real-life applications. For surveys of the theme one should read [126] and [52].

In [26] it is shown that, for continuous linear optimization problems, the gap between the adjustable and classical robust solutions are due to the fact that the classical robust formulation is inherently unable to model non-constraint-wise uncertainty and replaces any given uncertainty set U, with a potentially much larger uncertainty set. The adjustable robust optimization solution is, in fact, at least as good as the equivalent classical robust optimization solution. Different works have

shown that, for some classes of problems, the optimal solution of classical robust optimization equals the solution of the adaptive variant.

In [19] the authors show that for a linear optimization problem with constraint-wise uncertainty and a compact uncertainty set, the optimal solution of classical robust optimization and of the adaptive variant are equal. Their result is further extended in [33] where it is shown that for a specific class of problems with non-constraint-wise uncertainty, an optimal solution for the classical robust problem is also optimal for the adjustable variant. They show that the optimality of classical robust solutions depends on the geometrical properties of a transformation of the uncertainty set. They show that the classical robust solution is optimal if the transformation of U is convex. If U is a constraint-wise set, the transformation of U is convex.

In [81] the authors prove that if the uncertainty is constraint-wise and the uncertainty set is compact, then under two sets of conditions classical robust solutions are optimal for the corresponding adaptive robust problem: (i) if the problem is fixed-recourse (the definition is provided later in this section) and (ii) if the problem is convex with respect to the adjustable variables and concave with respect to the parameters defining constraint-wise uncertainty, the adjustable variables lie in a convex and compact set and the uncertainty set is convex.

In [81] they also remark that, for such above cases, there is no need to solve the adaptive robust optimization problem and this has two important merits. First, solving a classical robust optimization problem is computationally easier than solving an adaptive robust one: there is no need to add extra variables to determine optimal coefficients in decision rules, and even in a non fixed-recourse situation the final problem is linear in the uncertain parameters. Second, since adaptive robust optimization is an on-line approach, parts of the solution can only be implemented after knowing the values of the uncertain parameters. However, the classical robust optimization approach is an off-line one, and all preparations for implementing the solution can start promptly after having it solved.

Approximation bounds for the performance of classical robust solutions in two-stage and multi-stage linear optimization problems have also been studied in the literature. In [30] the authors show that if the uncertainty set is perfectly symmetric (such as a hypercube or an ellipsoid), a classical robust solution gives a 2-approximation for the adjustable robust linear covering problems under right-hand-side uncertainty. In [32] the authors generalize the result for general convex compact uncertainty sets and show that the performance of the classical robust solution depends on a measure of symmetry of the uncertainty set.

In [33] the authors also show that, for the classes of problems that they define, when a classical robust solution is not optimal for the adjustable robust problem,

there exists a tight approximation bound on the performance of the classical robust solution that is related to a measure of non-convexity of a transformation of the uncertainty set.

Due to its inherent capacity to reduce conservatism and simpler formulation, many applications have been modeled as two-stage robust optimization problems, where there is only two stages of decisions possible (see, for examples, [23, 57, 88]). For the sake of brevity, we consider an extra vector ξ, that represents the "primary" uncertainty, and define $d(\xi)$, $A(\xi)$, $B(\xi)$ as affine functions of ξ. For our linear robust problem this leads to a formulation as

$$\min_{x,y()} \left\{ \max_{\xi \in \Xi} c^T x + d(\xi)^T y(\xi) : A(\xi)x + B(\xi)y(\xi) \leq b, \forall\, \xi \in \Xi \right\}, \tag{2.6}$$

where $\xi \in \mathbb{R}^{n_2}$ varies within a known uncertainty set Ξ. Also, $x \in \mathbb{R}^n$ are first-stage decision variables, $y \in \mathbb{R}^{n_1}$ are second-stage decision variables and are dependent on the realization of uncertainty. We can assume without loss of generality that coefficient c of first-stage variables x are certain and $B(\xi)$ is called the recourse matrix. The objective function has two parts, reflecting the two-stage nature of the decision. From this formulation, we can see that the first-stage variables x takes into account all possible future variations represented in the uncertainty set. Such a solution remains feasible, thus robust, for any realization of the uncertainty set. Furthermore, in our formulation the optimal second-stage decision is a function of the uncertainty, therefore, fully adaptive to any realization of the uncertainty. Notice that is also a function of the first-stage decision. However, we write it as $y(\xi)$ to emphasize the adaptability of the second-stage decision to the uncertainty. The above formulation can be recast in the following equivalent form, which is suitable for developing numerical algorithms:

$$\min_{x} \left\{ c^T x + \max_{\xi \in \Xi} \min_{y \in \Omega(x,\xi)} d(\xi)^T y(\xi) \right\}, \tag{2.7}$$

where $\Omega(x, \xi) = \{y \mid By \leq b - Ax\}$.

Problem (2.6) is a complex one, that involves infinitely many decision variables and constraints. It has been shown to be NP-hard in general form unless we restrict $y(\xi)$ to specific decision rules classes that provides approximation solutions [68].

2.3.1 Exact solutions

To derive exact solutions for problem (2.7) two main strategies have been developed in the literature of robust optimization. The first strategy, developed initially in [121] for robust optimization problems, gradually constructs the value function of

the first-stage decisions using dual solutions of the second-stage decision problem. They are actually very similar to Benders decomposition method in that constraints that approximate the value function are iteratively generated from a subproblem and then supplied to a master problem. In [121] they verify that the method used will vary depending on the recourse matrix W and the recourse function defined as the inner problem and given by

$$Q(x, \xi) = \min_{y \in \Omega(x,\xi)} d^T y, \tag{2.8}$$

where they suppose that $Q(x, \xi)$ possesses relatively complete recourse, meaning it is feasible for all values of (x, ξ). They develop an initial method for fixed-recourse, when the recourse matrix W is deterministic, and only the right-hand-side vector b is uncertain. They also develop methods for the cases of simple recourse, in the case the recourse matrix W has the structure $W = [I \ -I]$ and I is the identity matrix, and analyze cases of general recourse.

The second strategy, defined in [128], present a constraint-and-column generation algorithm to solve two-stage robust optimization problems. It does not create constraints using dual solutions of the second-stage decision problem. Instead, it dynamically generates constraints with recourse decision variables for an identified scenario in the uncertainty set in each iteration, which is very different from the philosophy behind Benders-dual procedures.

Additionally, an alternative solution algorithm that exploits duality is derived for general two-stage adaptive linear optimization models with polyhedral uncertainty sets in [27]. Although they model the uncertainty as right-hand-sided, the same derivation can be done in case of fixed-recourse. The new model is obtained by consecutively dualizing over the wait-and-see decision variables and then over the uncertain parameters. The resulting problem is again an adaptive robust optimization problem, in which some of the dual variables are adjustable. Therefore, all existing solution approaches for two-stage adaptive models can be used to solve or approximate the dual formulation. The new dualized model differs from the primal formulation in its dimension and uses a different description of the uncertainty set. One advantage of the new dual adjustable model is that it is solved an order of magnitude faster than the primal adjustable formulation. If affine decision rules (see below) are adopted for the dual variables the resulting problem is tractable.

If we restrict $y(\xi)$ to specific decision rules classes, approximation algorithms can be developed. With this restriction, two-stage robust optimization formulations can be generally reduced to classical robust optimization problems.

2.3.2 Affine decision rules

For affine decision rules, second-stage variables y are defined as affine functions of the form:

$$y(\xi) = y^0 + Y^1\xi \tag{2.9}$$

where $y^0 \in \mathbb{R}^n$ and $Y^1 \in \mathbb{R}^{n_1 \times n_2}$. The work [19] was the first to propose using affine decision rules to approximate the adaptive robust problem. It was shown that affine decision rules result into tractable problems in the case of linear optimization with fixed-recourse. The resulting problem is linear in the optimization variables and in the uncertain parameters, and is therefore theoretically as tractable as the original robust optimization problem for any uncertainty set, although it may have many more variables. The optimal solution (x, y^0, Y^1) contains the here-and-now decision x that can immediately be implemented, and the coefficients for computing $y(\xi)$ as soon as ξ is observed. So, the solution dictates the second-stage decision without requiring a new optimization model to be solved.

In [81] the authors prove that for a specific class of problem and if it has both constraint-wise and not constraint-wise uncertain parameters, using affine decision rules that depend on both types of uncertain parameters yields the same optimal objective value as ones depending solely on the non-constraint-wise uncertain parameter. That reduces the number of variables in the problem used to model affine decision rules, because it is known beforehand that the coefficients of the constraint-wise uncertain parameters are zero.

The power of the approximation of affine decision rules has been object of many studies. In [24] they consider two-stage adjustable robust linear optimization problems with uncertain right-hand-side b belonging to a convex and compact uncertainty set U. They provide an a-priori approximation bound on the ratio of the optimal affine solution to the optimal adjustable solution that depends on two fundamental geometric properties of U: (a) the "symmetry" and (b) the "simplex dilation factor" of the uncertainty set U and provides deeper insight on the power of affine policies for this class of problems. In [29] they also consider a two-stage adaptive linear optimization problem under right-hand-side uncertainty and give a sharp characterization of the power and limitations of affine policies. In particular, they show that the worst-case cost of an optimal affine policy can be $\Omega(m^{\frac{1}{2}-\delta})$ times the worst-case cost of an optimal fully-adaptable solution for any $\delta > 0$, where m is the number of linear constraints. They also show that the worst-case cost of the best affine policy is $O(m^{-\frac{1}{2}})$ times the optimal cost when the first-stage constraint matrix has non-negative coefficients.

Many extensions of affine decision rules to more complex (non-linear) decision rules have been proposed in the literature, see the recent survey [126]. Among

them, affine decision rules defined over extended uncertainty sets offer good trade-offs between tractability and expressiveness [15, 45].

2.3.3 Finite adaptability

The framework introduced above does not work if the second-stage variables are subject to integer constraints, since the second-stage variables y are necessarily continuous functions of the uncertainty. Finite adaptability, introduced in [26], is designed to deal exactly with this setup. There, the second-stage variables, y, are piecewise constant functions of the uncertainty, with K pieces. Due to the inherent finiteness of the framework, the resulting formulation can accommodate discrete variables. In addition, the level of adaptability can be adjusted by changing the number of pieces K in the piecewise constant second-stage variables.

In the finite adaptability framework, with K-piecewise constant second-stage variables, our adaptive robust optimization problem becomes

$$\min_{x, y_{k\in\{1,\dots,K\}}} \left\{ \max_{\xi\in\Xi} c^T x + \min_{k\in\{1,\dots,K\}} d(\xi)^T y_k : A(\xi)x + B(\xi)y_k \leq b \right\}, \tag{2.10}$$

Problem (2.10) determines K non-adjustable decision rules $y_1, \dots, y_K$ here-and-now and subsequently selects the best of these decisions in response to the observed value of uncertainty. If all decisions are infeasible for some realization of uncertainty, then the solution $(x, y_{k\in\{1,\dots,K\}})$ attains the objective value $+\infty$. By construction, the K-adaptability problem (2.10) bounds the two-stage robust optimization problem from above.

In this single-stage formulation, the K-adaptability formulation can also have the interpretation of the decision-maker obtaining side-information about the uncertainty, before it is fully revealed. That is, we can equivalently consider the problem where the decision-maker selects the partition of Ξ into K regions, and then receives advance information that the uncertainty realization will fall in region i, $1 < i < K$. If the partition of the uncertainty set, $\Xi = \Xi_1 \cup, \cdots, \cup \Xi_K$ is fixed, then the resulting problem retains the structure of the original nominal problem, and the number of second-stage variables grows by a factor of K. Problem (2.10) can then be reformulated as

$$\begin{array}{lll} \min\limits_{x} & c^T x + t & \\ s.t. & A(\xi)x + B(\xi)y_k \leq b & \forall\, \xi \in \Xi_k, k \in \{1, \dots, K\} \\ & d(\xi)^T y_k \leq t & \forall\, \xi \in \Xi_k, k \in \{1, \dots, K\} \end{array}$$

The complexity of finite adaptability is in finding a good partition of the uncer-

tainty. In [26] they verify that, in general, computing the optimal partition even into two regions is NP-hard. There they reformulate the two-adaptability problem as a finite-dimensional bilinear program and solve it heuristically.

The relationship between the K-adaptability problem (2.10) and classical robust optimization is further explored in [32] for the special case where matrices A and B are deterministic. The authors show that the gaps between both problems are intimately related to geometric properties of the uncertainty set Ξ.

Finite-dimensional MILP reformulations for problem (2.10) are developed in [69] under the additional assumption that both the here-and-now decisions x and the wait-and-see decisions y are binary. The authors show that both the size of the reformulations as well as their gaps to the two-stage robust optimization problem (1) depend on whether the uncertainty only affects the objective coefficients d, or whether the constraint coefficients A, B and b are uncertain as well.

In [116] two-stage robust optimization problems with mixed discrete-continuous decisions in both stages are studied. The authors compare the two-stage robust optimization problem with the K-adaptability problem in terms of their continuity, convexity and tractability. They investigate when the approximation offered by the K-adaptability problem is tight, and under which conditions the two-stage robust optimization and K-adaptability problems reduce to single-stage problems. They develop a branch-and-bound algorithm for the K-adaptability problem that combines ideas from semi-infinite and disjunctive programming and present a heuristic variant that can address large-scale instances.

2.3.4 Nonlinear decision rules

Many other different classes of decision rules have been studied in the literature. In [126] the authors present a summary of different classes of nonlinear decision rules and the resulting complexity of the equivalent adaptive robust optimization problem incorporating the decision rules.

2.4 Objective function

Another potential source of over conservatism in classical robust optimization is the objective function that is guaranteed to yield to best performance only when worst possible realization of uncertainty has occurred. The fact that this solution is suboptimal when the worst-case realization does not occur can make it unpractical for many applications.

Different modeling techniques have been developed to deal with this issue. They trade some of the robustness in exchange for potential better objective performance.

In [63] the authors develop a model they call Light Robustness. In this model they introduce a constraint to fix the worst objective function deterioration they are willing to accept. This deterioration is calculated in relation to the solution considering only nominal values of uncertain parameters. To handle possible infeasibility they create slack variables, as second-stage variables, that allow local violations of the robustness requirements and define an auxiliary objective function aimed at minimizing the slacks.

Other authors have explored concepts of multi-objective optimization and Pareto efficiency to develop less conservative models. A solution is called Pareto optimal if there exists no other solution with performance in every objective function at least as good as its performance and is strictly better in at least one objective function.

In [42] the authors propose to include the average-case performance as an objective function, thus resulting in a bicriteria problem. They remark that a frequently used assumption is that the objective function of the optimization problem is certain, as every objective function can be represented as a constraint by using the epigraph transformation. While this is a valid method, it has the drawback that feasibility and performance guarantees are mixed into one criterion, which is questionable for most practical problems. Therefore, they focus explicitly on problems that are affected by uncertainty only in the objective function and assume that the set of feasible solutions is restricted to solutions that are feasible for all possible parameter values. They define the bicriteria optimization problem with the two objective functions average and worst-case performance, and the set of optimal solutions of this problem as the average-case to worst-case curve (AC–WC curve). To compute the AC–WC curve, they make algorithmic use of the observation that a robust solution can be interpreted as a special point on the Pareto curve with respect to a multi-criteria problem where every possible scenario outcome leads to its own objective. They build upon a theoretical observation on the structure of Pareto solutions for problems with polyhedral feasible sets and present a column generation approach that does not require direct solution of the worst-case problem.

In [71] the authors define a solution to be Pareto robustly optimal when its performance in one scenario cannot be improved without worsening its performance in another scenario. The authors focus on the robust optimization linear problem where only the objective function coefficients, c, are uncertain. The authors propose to solve an additional linear program of compact size to determine if the solution of the original robust problem is Pareto optimal. If not, the solution of this linear program provides a complement to transform the original solution into a Pareto optimal solution. The associated approach can be extended to a more general class of optimization problems where matrix A and b of our robust problem are uncertain.

In [51] the authors show that adaptive robust optimization yields multiple ro-

bust optimal solutions for a production- inventory example and show that different adaptive robust solutions that have the same worst-case objective function value differ up to 21.9% in the mean objective value. They deduce three important implications of the existence of multiple optimal adaptive robust solutions. First, if one neglects this existence of multiple solutions, then one can wrongly conclude that the adjustable robust solution does not outperform the nonadjustable robust solution. Second, even when it is a-priori known that the adjustable and nonadjustable robust solutions are equivalent on worst-case objective value, they might still differ on the mean objective value. Third, even if it is known that affine decision rules yield (near) optimal performance in the adjustable robust optimization setting, then still nonlinear decision rules can yield much better mean objective values. The aim is to convince users of adjustable robust optimization to check for existence of multiple solutions. They apply a two-step approach to find a Pareto robust optimal solution among the alternative adjustable robust optimal solutions.

These alternatives do not consider the stochastic nature of uncertainty and do not exploit additional distributional information. Distributionally Robust Optimization, which dates back to the work of [105] is a less conservative alternative that considers that the probability distributions of uncertain parameters are assumed to belong to an ambiguity set, family of distributions that share common properties. It is a robust formulation for stochastic programming problems. The concept was used as extension to min-max stochastic optimization models (see, for example, [109]), and recently to distributionally robust optimization models (see, for instance, [46, 53, 66]).

In this model, after defining a set $\mathcal{D}$ of probability distributions that is assumed to include the true distribution $\mathbb{P}$, the objective function is reformulated with respect to the worst-case expected cost over the choice of a distribution in this set. Hence, this leads to solving the Distributionally Robust Stochastic Program

$$\min_{x \in X} \max_{\mathbb{P} \in \mathcal{D}} \mathbb{E}_{\mathbb{P}}[Q(x, \xi)], \tag{2.11}$$

where $Q(x, \xi)$ is a convex cost function in x that depends on some vector of random parameters ξ and $\mathbb{E}_{\mathbb{P}}$ is the expectation taken with respect to the random vector ξ given that it follows the probability distribution $\mathbb{P}$.

Since the introduction of distributionally robust optimization, several ambiguity sets have been proposed and analyzed. Among these, three types have received significant attention: moment-based ambiguity sets, structural ambiguity sets, and metric-based ambiguity sets.

In moment-based ambiguity sets, it is assumed that all distributions in the distribution family share the same moment information. Leveraging conic duality for

moment problems (see [110]) and developments in interior point algorithms for solving semidefinite programming problems, many distributionally robust optimization problems with moment-based ambiguity sets can be reformulated equivalently (or approximately) as semidefinite programming problems. In [53], for example, the authors study tractable reformulations for distributionally robust expectation constraints under the assumption that the ambiguity set specifies the support as well as conic uncertainty sets for the mean and the covariance matrix of the uncertain parameters. The authors also provide a recipe for constructing ambiguity sets from historical data using McDiarmid's inequality.

In structural ambiguity sets, distributions share the same structural properties, such as symmetry, unimodality, and monotonicity. In [122], for example, the authors study worst-case probability and conditional value-at-risk problems, where the distributional information is limited to second-order moment information in conjunction with structural information such as unimodality and monotonicity of the distributions involved. They indicate how exact and tractable convex reformulations can be obtained using tools from convex analysis, more specifically Choquet and duality theory.

Metric-based ambiguity sets are created by requiring that all distributions are close to a reference (or nominal) distribution within a prespecified probability distance. The reference distribution is usually estimated using sampled data. By adjusting the radius of the ambiguity set, the modeler can thus control the degree of conservatism of the underlying optimization problem. If the radius drops to zero, then the ambiguity set shrinks to a singleton that contains only the nominal distribution, in which case the distributionally robust problem reduces to an ambiguity-free stochastic program. In addition, ambiguity sets can also be defined as confidence regions of goodness-of-fit tests. In [86], for example, using the Wasserstein metric (see [115] for a definition), the authors construct a ball in the space of (multivariate and non-discrete) probability distributions centered at the uniform distribution on the data training samples, and they seek decisions that perform best in view of the worst-case distribution within this Wasserstein ball. They show that, under mild assumptions, the distributionally robust optimization problems over Wasserstein balls can in fact be reformulated as finite convex programs.

Chapter 3

Solving the bifurcated and non-bifurcated robust network loading problem with k-adaptive routing

3.1 Introduction

This study is about a robust linear optimization approach for the network loading problem under demand uncertainty (RNL). Given a graph $G(V, E)$ and a set of node-to-node uncertain demands as commodity origin-destination flows, we want to define minimum cost integer capacity installations for the edges (investment decisions), such that all commodities can be routed simultaneously on the network (routing decisions defining a routing scheme).

Robust network design problems, in general, have been widely studied motivated by the fact that demands in modern applications vary and are hard to forecast. The standard robust approach encompasses solutions that are feasible for any realization of uncertainty, so that a decision taken hedges against the worst contingency that may arise. This leads to a conservative solution approach.

A robust approach to the network design problem with demand uncertainty can be tracked back to the work of [80]. They introduce a polyhedral uncertainty set for demands and a min-max-min approach to the problem using dualizations to derive an equivalent linear problem formulation. In [77] the authors analyse for the first time many network design problems from a robustness perspective and introduce the term robust network design. They assert that multicommodity network design problems exhibit the important property of having multiple, and significantly different, solutions within a few percentage points from the optimal. This feature makes

them amenable to algorithmic developments that can find common near optimal (robust) solutions for a variety of future operating scenarios. The uncertainty in the problems they consider are in the costs of the networks and are modeled as discrete scenarios.

Modeling uncertainty as discrete scenarios is limiting (see [44]) and since then many studies have been made on how to model uncertainty and the effect this has on the formulation, solution, and complexity of algorithms developed. Modeling uncertainty has a objective of reflecting real data behavior, but can affect conservatism of the solution.

Conservatism is also related to how the demands are routed through the network, meaning how commodity flows are carried from source to destination using network resources. The problem can be seen as a two stage robust optimization problem where the recourse function consists in choosing the routing for each demand vector. In fact, robust network design problems have an intrinsic separation between first stage decisions and second stage decisions: investment decisions must be made before we observe the results of the demand uncertainty, while routing decisions have to route whatever demand occurred. One can decide which route to take based on actual demands, leading to less conservative solutions.

Routes can also have restrictions inherent to the application being modeled. Possible restrictions are if demands can be split between different paths (bifurcated) or not (non-bifurcated) or if there is a maximum number of edges to be used for each path, among others.

In this study we experiment and compare different modeling strategies to treat conservatism of robust solutions for RNL.

The purpose of this study, which is an extension of the work presented in the short paper [113], is four-fold:

- We present a review of the literature on robust network design, focusing on the polyhedral uncertainty sets and different routings used.

- We show the application of the iterative k-adaptive partitioning algorithm defined in [28] to RNL. We develop and apply decomposition techniques to this application and we verify its time performance comparing against an implementation using the recent Benders Decomposition functions of CPLEX 12.7.

- For non-bifurcated flows, we show the cost reductions provided by k-adaptive routing scheme over static. This is the first attempt to improve over static solutions for non-bifurcated flows. So far, solutions developed assume continuous second stage decision variables. These previous approaches include

enumerating the extreme points of the uncertainty set (see [98]), using classical decomposition algorithms (see [8, 83]) or using affine decision rules (see [93, 94, 98]). Integrality of the second stage variables prevents us from using these approaches.

- For bifurcated flows, we compare the solution times and costs of the k-adaptive routing scheme with those of static, volume and two-partitioning routings.

Outline We first review in Section 3.2 the literature dealing with polyhedral uncertainty sets and routing schemes, applied to robust network design problems. Next, in Section 3.3, we formally present the problem that we will focus on our experiment. Section 3.4 details key concepts used in our routing scheme, that dynamically partitions the uncertainty set. In Section 3.5 we explain the solution strategy, with emphasis on techniques to reduce branching and decomposition techniques developed to cope with large instances. In Section 3.6 we present algorithm implementation details and results.

3.2 Literature review

3.2.1 Polyhedral uncertainty set

The authors of [13] model demand uncertainty with a polyhedral uncertainty set and argue that the polytope so defined, D, has to be sufficiently large to allow routes to be flexible, but not excessively large to avoid wasting network resources. Each demand vector is defined in its components by its node origin and destination (d_{ij}, $i, j \in V$). They define a model in which A is a real-valued matrix and b is a real-valued vector and

$$D = \left\{ d \in \mathbb{R}_+^{|V| \text{ x } |V-1|} \mid Ad \leq b \right\}$$

Prior to that, another approach in the same direction can be found in the work of [62] for ATM's communications network and [59] for VPN's communications networks. They introduced a particular polyhedral model for uncertainty, based on the definition of demand upper bounds, named the Hose model. In the Hose model two upper bound demand parameters are defined for each node $i \in V$: d_i^{out}, as outgoing demand from node i and d_i^{in}, as ingoing demand to node i:

$$D = \left\{ d \in \mathbb{R}_+^{|V| \text{ x } |V-1|} \mid \sum_{j \in V \setminus \{i\}} d_{ij} \leq d_i^{out}, \sum_{j \in V \setminus \{i\}} d_{ji} \leq d_i^{in}, \quad \forall i \in V \right\}$$

A breakthrough for the design of polyhedral uncertainty sets that influenced fu-

ture work on network design was in [31]. Therein the authors protect against the violation of constraint i of an optimization problem deterministically, when only a pre-specified number Γ_i of the coefficients change and where each coefficient is defined within a band of variation. In our demand uncertainty context this band defines a variation around a given nominal demand $\bar{d}_{ij}$ with a maximum possible deviation value $\hat{d}_{ij} \geq 0$. It is named the gamma-model or budgeted uncertainty model. Considering for our purposes only positive deviations, the equivalent polyhedral uncertainty polytope that results is defined by:

$$D = \left\{ d \in \mathbb{R}_{+}^{|V| \text{ x } |V-1|} \mid \bar{d}_{ij} \leq d_{ij} \leq \bar{d}_{ij} + \hat{d}_{ij}; \sum_{j \in V, j \neq i} \frac{d_{ij} - \bar{d}_{ij}}{\hat{d}_{ij}} \leq \Gamma_i; \, i, j \in V \right\}$$

Further studies on polyhedral uncertainty sets have identified other opportunities to reduce conservatism. In [39] the authors verify that the single deviation band for each coefficient proposed in [31] may be too limited in practice, so that getting a higher resolution by partitioning the band into multiple sub-bands seems advisable (see also [49]). In [97], the author introduces the concept of decision dependent uncertainty as an extension of the work presented in [31]. The author proposes a new model where the uncertain parameters belong to the image of multi-functions of the problem decision variables. It can provide the same probabilistic guarantee as the uncertainty set defined in [31] while being less conservative. The model has been revived more recently in [90].

3.2.2 Routing

We define each routing scheme from a robust optimization perspective as in [96]. Therein, a formal definition of routing is made. A routing is a function $f : D \subset \mathbb{R}^{|Q|} \rightarrow \mathbb{R}^{|E| \text{ x } |Q|}$ that associates each demand vector, expressed by its components (d^q) as commodities $(q \in Q)$ with nodes origin $(s(q))$ and destination $(t(q))$, with a multicommodity flow, defined for each edge and commodity. This function satisfies the flow conservation constraint at each node of the network, where $\delta(v) = \{ij \in E \quad | \quad i = v \text{ or } j = v\}$ and

$$\sum_{j \in \delta(i)} (f_{ij}^q(d) - f_{ji}^q(d)) = \begin{cases} d^q & \text{if } i = s(q) \\ -d^q & \text{if } i = t(q), \text{ for each } i \in V \\ 0 & \text{else} \end{cases} \qquad \forall q \in Q \tag{3.1}$$

$$f_{ij}^q(d) \geq 0 \quad \forall ij \in E, \forall q \in Q \tag{3.2}$$

One routing scheme, called dynamic routing, is the set of all functions that satisfy

(3.1) and (3.2) :

$$\mathbb{F} \equiv \{f : D \to \mathbb{R}^{|E| \text{ x } |Q|} \quad | \quad f \text{ satisfies (3.1) and (3.2)}\}$$

Named dynamic routing, permits full flexibility in re-routing according to demand changes and, in consequence, is potentially less costly. It has been shown in [43] that the robust network design problem with dynamic routing is NP-hard, and in [85] that it is also NP-hard for polyhedral uncertainty sets.

An opposite approach to dynamic routing has been introduced by [13] for the robust network design problem. Designated static routing, meaning that for every commodity the same paths are used with the same splitting independently of the realization of demand. It amounts to restricting the second stage recourse. Each function component $f^q : D$ to $\mathbb{R}_+^{|E|}$ is a linear function of d^q :

$$f_{ij}^q(d) := y_{ij}^q d^q, ij \in E, q \in Q, d \in D \tag{3.3}$$

The flow y is called a routing template since it decides, for every commodity, which paths are used to route the demand and what is the percentage splitting among these paths. Combining equations (3.1) and (3.3), the routing template y satisfies

$$\sum_{j \in \delta(i)} (y_{ij}^q - y_{ji}^q) = \begin{cases} 1 & \text{if } i = s(q) \\ -1 & \text{if } i = t(q), \text{ for each } i \in V \\ 0 & \text{else} \end{cases} \qquad \forall q \in Q \tag{3.4}$$

We define formally the set of all routing templates:

$$\mathbb{Y} \equiv \{y \in \mathbb{R}_+^{|E| \text{ x } |Q|} \quad | \quad y \quad \text{satisfies (3.4)}\} \tag{3.5}$$

With that, we can define the set of all static routings:

$$\mathbb{F}^{stat} \equiv \{f : D \to \mathbb{R}^{|E| \text{ x } |Q|} \quad | \quad \exists y \in \mathbb{Y} : \quad f \text{ satisfies (3.3)}\}$$

The resulting optimization problem is polynomially solvable.

Recently significant progress has been made in defining routing schemes in between static and dynamic, imposing a restriction on how the flow can be tuned to demand.

Defined in [20] as the simplest restriction of this type (two stage), affine routing requires the recourse variables to be affine functions of the data. The affine approximation assumes continuous second stage variables and therefore does not support second stage integer variables, or unsplittable flows for RNL. Affine routing was

first addressed for the robust network design problem by [93]. They did that by restricting f^q to be an affine function of all components of d giving

$$\mathbb{F}^{aff} \equiv \{f : D \to \mathbb{R}^{|E| \text{ x } |Q|} \quad | \quad \exists f_0 \in \mathbb{R}^{|E| \text{ x } |Q|}, y \in \mathbb{R}^{|E| \text{ x } |Q| \text{ x } |Q|} : \\ f_{ij}^q(d) = f_{ij}^{0q} + \sum_{h \in Q} y_{ij}^{qh} d^h, \quad ij \in E, q \in Q, d \in D, \text{ and } f \text{ satisfies (3.1) and (3.2)}\}$$

In [84] the authors perform a numerical study of the affine routing robust network design problem. They show that formulations tend to be so large that they become hard to solve for large instances. In this context the authors suggest that it might be wise to restrict the number of commodities in the affine recourse or apply decomposition methods.

Another concept for routing is derived from the idea of sharing two routes for each commodity in a way that depends on the actual volumes of demands. This type of routing, called volume routing, was originally introduced in [130]. Formally, the authors use the following set of routings, according to thresholds h^q for each $q \in Q$:

$$\mathbb{F}^V \equiv \{f : D \to \mathbb{R}^{|E| \text{ x } |Q|} \quad | \quad \exists y^1, y^2 \in \mathbb{Y}, h \in \mathbb{R}_+^{|Q|} : \\ f_{ij}^q(d) = y_{ij}^{1q} \min(d^q, h^q) + y_{ij}^{2q} \max(d^q - h^q, 0) \quad ij \in E, q \in Q, d \in D\}$$

They prove that the robust network design problem associated with $\mathbb{F}^V$ is an NP-hard optimization problem. Hence, they introduce simpler frameworks described below. Defining $d_{min}^q = \min_{d \in D} d^q$ and $d_{max}^q = \max_{d \in D} d^q$, the set of routings becomes one of the following:

$$\mathbb{F}^{VS} \equiv \{f : D \to \mathbb{R}^{|E| \text{ x } |Q|} \quad | \quad \exists y^1, y^2 \in \mathbb{Y} : \\ f_{ij}^q(d) = y_{ij}^{1q} d_{min}^q + y_{ij}^{2q}(d^q - d_{min}^q), \quad ij \in E, q \in Q, d \in D\}$$

$$\mathbb{F}^{VG} \equiv \{f : D \to \mathbb{R}^{|E| \text{ x } |Q|} \quad | \quad \exists y^1, y^2 \in \mathbb{Y} : \\ f_{ij}^q(d) = y_{ij}^{1q} d_{min}^q \frac{d_{max}^q - d^q}{d_{max}^q - d_{min}^q} + y_{ij}^{2q} d_{max}^q \frac{d^q - d_{min}^q}{d_{max}^q - d_{min}^q}, \\ ij \in E, q \in Q, d \in D\}$$

which are both well defined whenever $d_{min}^q < d_{max}^q$ for each $q \in Q$. When $d_{min}^q = d_{max}^q$ for some $q \in Q$, the q-th component of $f \in \mathbb{F}^{VG}$ is defined by $f^q(d) = y^{1q} d^q$. By definition $\mathbb{F}^{VS}$ and $\mathbb{F}^{VG}$ are special cases of affine routing.

A generic volume routing is defined in [84], where each commodity flow is based on two components: a flow that varies with demand and is a set of arc-paths from origin to destination plus a circulation that is independent of the demand value (see [98] for details). It also restricts the number of commodities in the affine recourse

and leads to smaller formulations.

$$\mathbb{F}^{VA} \equiv \{f : D \to \mathbb{R}^{|E| \text{ x } |Q|} \quad | \quad \exists f_0 \in \mathbb{R}^{|E| \text{ x } |Q|}, y \in \mathbb{Y} : \\ f_{ij}^q(d) = f_{ij}^{0q} + y_{ij}^q d^q, \quad ij \in E, q \in Q, d \in D, \text{ } f \text{ satisfies (3.1) and (3.2)}\}$$

In [96] the author compares optimal solutions for different routing strategies F, defined as $opt(F)$. They show that $opt(\mathbb{F}^{aff}) \leq opt(\mathbb{F}^{VG}) \leq opt(\mathbb{F}^{VS})$, and $opt(\mathbb{F}^{V}) \leq opt(\mathbb{F}^{VS})$. They also show that it is not possible, in general, to order $opt(\mathbb{F}^{V})$ and $opt(\mathbb{F}^{aff})$. Also one can check that $opt(\mathbb{F}^{aff}) \leq opt(\mathbb{F}^{VA}) \leq opt(\mathbb{F}^{VG})$ using the same arguments of [96].

More recently, in [14], an alternative routing scheme is proposed with the objective of maintaining tractability of the reformulated optimization problem to be solved. It assumes a polyhedral uncertainty set and fixed recourse. Called multipolar routing, the authors approximate the uncertainty set by a polytope defined by its extreme routings, named poles. Each pole $k \in \mathbb{K}$ is a special demand vector, not necessarily belonging to D. The poles are given beforehand and are defined so that

$$\exists \lambda \mid \sum_{k \in \mathbb{K}} \lambda_d^k k^q = d^q; \quad \sum_{k \in \mathbb{K}} \lambda_d^k \leq 1; \quad \lambda_d^k \geq 0; \quad \forall d \in D, \forall q \in Q \tag{3.6}$$

Each pole k is defined as having a route associated with it, so that we have

$$f_{ij}^q(k) = y_{ij}^{qk} k^q, \quad ij \in E, q \in Q, k \in \mathbb{K} \tag{3.7}$$

Moreover, the route for each realization of the demand vector will be a convex combination of the routes defined for each pole. Expression (3.6) states also how each demand should be routed and we define

$$f_{ij}^q(d) = \sum_{k \in \mathbb{K}} \lambda_d^k y_{ij}^{qk} k^q, ij \in E, q \in Q, k \in \mathbb{K} \tag{3.8}$$

We define formally the set of all multipolar routings as

$$\mathbb{F}^{MP} \equiv \{f : D \to \mathbb{R}^{|E| \text{ x } |Q|} | \quad \exists \lambda \in \mathbb{R}^{|\mathbb{K}| \text{ x } |D|}, y \in \mathbb{R}^{|\mathbb{K}| \text{ x } |E| \text{ x } |Q|}, \\ f \text{ satisfies (3.1), (3.2), (3.8)}\} \tag{3.9}$$

One recurring idea for another approximate routing is the partitioning of the uncertainty set into subsets and considering a static routing for each subset. It has been studied under the general framework of multi-stage robust optimization in [26]. Therein, it is called finite adaptability, where the decision-maker chooses k second-stage static solutions, and then commits to one of them only after seeing the realization of the uncertainty. The decision-maker defines a cover of the uncertainty

set with k subsets (possibly non-disjoint). It has the advantage of being able to naturally accommodate discrete second stage decision variables. We will call this routing scheme k-adaptive ($\mathbb{F}^k$).

The inequalities $opt(\mathbb{F}) \leq opt(\mathbb{F}^k) \leq opt(\mathbb{F}^{stat})$ hold in general. The authors provide an important geometric interpretation of the gap between static and dynamic routing solutions. They show that the gap is related to the fact that the static solution is not able to correlate uncertainty components between different constraints. For polyhedral uncertainty sets they show that this gap can be expressed in terms of matrices that belong to the uncertainty set. The authors also affirm that their k-adaptability proposal is not comparable to affine adaptability: in some cases affine adaptability fails where k-adaptability succeeds, and vice versa, so that we cannot order conservatism between the two approaches. The quality of solutions obtained with a k-adaptive approach depends on how the partition is built. Therefore there is a natural trade-off between the number of sets of a partition (and the solution time) and how "close" the solution is to the dynamic solution. Hence, a goal when using k-adaptability is to identify a partitioning scheme that is near the efficient frontier of this trade-off. In [26], the authors show that even finding an optimal 2 partitioning is NP-hard and reformulate it as a bilinear optimization problem. Being NP-hard, many heuristics have been proposed on how to split the uncertainty set.

In [12], the author introduces the idea of partitioning the uncertainty set with two (or more) subsets using hyperplanes and proposes to use a static routing template for each subset. This yields the following set of routings:

$$\mathbb{F}^{2|} \equiv \{f : D \to \mathbb{R}^{|E| \text{ x } |Q|} \quad | \quad \exists y^1, y^2 \in \mathbb{Y} \text{ and } \alpha \in \mathbb{R}^q, \beta \in \mathbb{R} :$$
$$f_{ij}^q = \begin{cases} y_{ij}^{1q} d^q & d \in D \cap \{d, \alpha d \leq \beta\} \\ y_{ij}^{2q} d^q & d \in D \cap \{d, \alpha d \geq \beta\} \end{cases}, \quad ij \in E, q \in Q, d \in D\}$$

The definition above implies that both routing templates y^1 and y^2 must be able to route demand vectors that lie in the hyperplane $\alpha d = \beta$ without exceeding the capacity. They prove that the problem is NP-hard in general and describe simplification schemes, where the direction α is given and solutions for each partition can be different or have to be identical.

The authors of [129] develop an algorithm based on a sequence of binary searches to solve a 2-partition robust linear problem with given hyperplane direction α and identical solutions. They define a routine that can answer the question: Is there a feasible solution with smaller or equal cost than a given value? They perform a binary search using this routine and given upper and lower bounds on the cost function. The routine is also based on binary searches using given β_{max} and β_{min} as bounds to constantly shrink the interval of possible positions of the hyperplane.

Also using the strategy of partitioning of the uncertainty set, [106] introduces the idea of using conjointly two routing templates. Formally, the author proposes to use two routing templates y^1 and y^2 such that each $d \in D$ can be served either by y^1 or by y^2 (or both). This yields the following set of routings:

$$\mathbb{F}^2 \equiv \{f : D \to \mathbb{R}^{|E| \text{ x } |Q|} \quad | \quad \exists y^1, y^2 \in \mathbb{Y} \text{ and } D^1, D^2 \subseteq D, D = D^1 \cup D^2 :$$
$$f_{ij}^q(d) = \begin{cases} y_{ij}^{1q} d^q & d \in D^1 \\ y_{ij}^{1q} d^q & d \in D^2 \end{cases} \quad ij \in E, q \in Q\}$$

3.3 Problem definition

RNL are robust network design problems with integer capacity decision variables, reflecting practical cases where a number of different facility type can be installed at each edge. We focus on a simplification of RNL where each commodity q can be routed along a predetermined set of paths $P(q)$. In many practical applications there are limits on the number of different configurations of paths that can be implemented, so that we exercise the flexibility of routing schemes to mitigate the static solution conservatism even with this restriction. It has been used in several papers, see for instance [94] and [93]. These paths are represented by δ_{qp}^{ij} that is equal to 1 if edge ij is contained in path $p \in P(q)$, for some $q \in Q$, and is equal to 0 otherwise. They were predetermined as shortest paths weighted by edges costs for each commodity.

We work with two cases: one in which flows are unsplittable, or non-bifurcated, and must use a single path, and another one in which flows are splittable, or bifurcated, and can be fractionally split among several paths. The cost for routing flows is zero.

Each commodity $q \in Q$ is associated with the uncertain demand d^q, within a given polyhedral uncertainty set D.

The formulation contains integer investment decision variables x, where x_{ij} equals the planned installed capacity for edge ij, considering only one type of facility at a unit cost of c_{ij}. It also contains continuous (for bifurcated flows) or binary (for non-bifurcated flows) routing decision variables y_{qp}, where y_{qp} equals the fraction of commodity $q \in Q$ assigned to path $p \in P(q)$.

Given a partition $D_1 \cup \cdots \cup D_K$ of the uncertainty set D, the k-adaptive routing scheme restricts the routing functions to piece-wise constant functions defined as $f_{qp}(d) = y_{qp}^k$ for $d \in D_k$ where $k \in \{1, ..., K\}$. The formulation follows, where $Y \equiv \{0, 1\}$ for non-bifurcated flows and $Y \equiv [0, 1] \subset \mathbb{R}$ for bifurcated flows and the

partition can be optimized along with the other optimization variables:

$$
\begin{aligned}
(kRNL) \qquad \min \quad & \sum_{ij \in E} c_{ij} x_{ij} & & \\
\text{s.t.} \quad & \sum_{q \in Q} \sum_{p \in P(q)} d^q \delta^{ij}_{qp} y^k_{qp} \leq x_{ij} & & ij \in E, \forall d \in D_k, \\
& & & k \in \{1, ..., K\} \qquad (3.10) \\
& \sum_{p \in P(q)} y^k_{qp} = 1 & & q \in Q, k \in \{1, ..., K\} \\
& y^k_{qp} \in Y, x_{ij} \geq 0, x_{ij} \in \mathbb{Z} & & q \in Q, p \in P(q), \\
& & & ij \in E, \\
& & & k \in \{1, ..., K\}
\end{aligned}
$$

As mentioned before, choosing the optimal partition makes the problem extremely difficult so that we focus on the heuristic solution algorithm proposed by [28].

3.4 Iterative nested partitioning

In [28] and [99] the authors independently identify that there is a set of active uncertain parameters $\hat{d}$. The active uncertain parameters are the values of the uncertain parameters that correspond to the constraints with minimum slack and that are the constraints that restricts the objective function. Applied to $kRNL$, the only constraints that involve uncertainty are constraints (3.10). We can define the set $\hat{D}$, the active uncertain parameters of D (or equivalently for any subset D_k), as follows. Given a solution $(\tilde{x}_{ij}, \tilde{y}_{qp})$ we define

$$
\hat{D} = \left\{ \hat{d} \mid \exists ij \in E \text{ s.t. } \hat{d} = \arg\min_{d \in D} (\tilde{x}_{ij} - \sum_{q \in Q} \sum_{p \in P(q)} d^q \delta^{ij}_{qp} \tilde{y}_{qp}) \right\} \qquad (3.11)
$$

where only one nominal element will be selected for each constraint involved in (3.11).

The authors in [28] identify that less conservatism can be obtained if a partition of the uncertainty set is created in such a way to guarantee that the active uncertain parameters do not all lie in one set of the partition and they construct a particular partition using Voronoi diagrams. Given a set of active uncertain parameters, the Voronoi diagram associated with these parameters defines a partition of D with one subset defined for each $\hat{d}_k \in \hat{D}$, such that the Euclidean distance between $\hat{d}_k$ and any d in this subset is less than or equal to the distance of d to any other given parameter $\hat{d}_j, j \neq k$:

$$D_k = D(\hat{d}_k) \quad = \{d \mid ||\hat{d}_k - d||_2 \leq ||\hat{d}_j - d||_2, \quad \forall \hat{d}_j, j \neq k\} \cap D \tag{3.12}$$

They further create the concept of nested partitions, where partitions can be created after previous partitions, so as to create a partition tree of active uncertain parameters. Voronoi diagrams are used to partition subsets of the previous level of the tree. The nodes of the tree, T, correspond to the set of all active uncertain parameters. Each level of the tree defines a sequence of subsets that define a partition of D at that level. To define these subsets under the concept of nested partitioning we need first to define some sets:

- $Leaves(T)$ is the set of leaves of the tree T. This is the set of all active uncertain parameters ($\hat{d}_k$) at the last level of the tree. As mentioned above, each one is uniquely associated with a subset of a partition being considered for the $kRNL$ problem. We index each set as $k \in \{1, ..., K\}$ as pointed out in our formulation of $kRNL$.

- $Children(\hat{d}_k)$ is the set of children of $\hat{d}_k$ in the tree T. This is the set of the active uncertain parameters associated with the partition of one subset.

- $Parent(\hat{d}_k)$ is the parent of $\hat{d}_k$ in the tree T. This is a one-element set defined by the active uncertain parameter that originated one subset at previous level of the tree.

- $Siblings(\hat{d}_k) = Children(Parent(\hat{d}_k))$

Hence a subset of the k-adaptive partition of the uncertainty set is obtained through a sequence of partitioning of many levels. Each subset is defined by the following inequalities:

$$\begin{aligned} D_k = D(\hat{d}_k) = \{d \mid ||\hat{d}_k - d||_2 \leq ||\hat{d}_j - d||_2, \quad \forall \hat{d}_j \in Siblings(\hat{d}_k)\} \\ \cap \quad \{d \mid ||Parent(\hat{d}_k) - d||_2 \leq ||\hat{d}_j - d||_2, \quad \forall \hat{d}_j \in Siblings(Parent(\hat{d}_k))\} \\ \cdots \\ \cap \quad D \end{aligned} \tag{3.13}$$

Figure 3.1 helps to explain the partitioning of equation (3.13). Note that each subset at the right hand side of equation (3.13) represents a subset of a partition at a specific level of the tree, and the union of subsets in the equation guarantees the nesting of partitioning. To facilitate the formulations derived in further sections,

we will represent the $M^{'}$ inequalities of polyhedral uncertainty set D_k by matrix notation $A_k^{M^{'} \text{ x } |Q|} d \leq b_k^{M^{'}}$.

An iterative method is built around the partitioning scheme. It starts by solving a first version of a $kRNL$ problem, where only the original uncertainty set is defined. The solution can be used to determine a set of active uncertain parameters. These are used, through Voronoi Diagrams and further partition of the uncertainty set, to construct another finitely-adaptive $kRNL$ version of the problem. It is solved. This in turn produces a new set of active uncertain parameters which we can then use to partition further, ideally improving on the previous solution at each iteration. These iterations can proceed until a termination criteria limit is encountered.

With this approach optimal solutions for a partition can be used as initial feasible solutions for the nested partitions originated from it. Each iteration will have a solution that is as good as or better than the previous iteration.

The pseudo code in Algorithm 1 reflects these main steps and defines the algorithm implemented in our experiment for the k-adaptive routing. As termination criteria we utilize a time limit procedure and a bounding procedure that will be explained in the next section (see below).

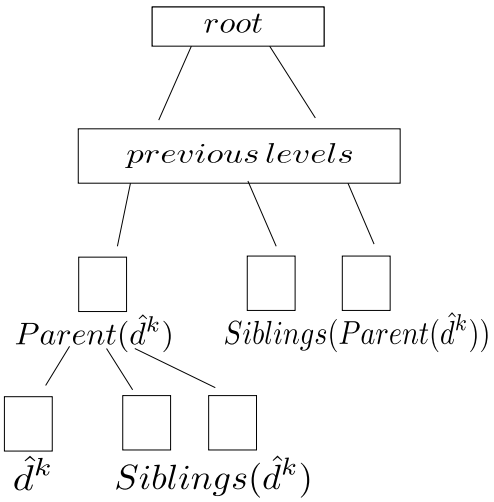

Figure 3.1: Nested partitioning

Algorithm 1 Iterative k-adaptive routing

```
Initialize                                          ▷ No partition at first round
repeat
    Solve kRNL Problem
    Calculate Bound : Termination Criteria
    if Bound gap limit not reached then
        Calculate Active Uncertain Parameters
        Formulate Nested Partitioning
    end if
until Time limit or Bound gap limit reached : Termination Criteria
```

3.5 Algorithm improvements

We describe below improvements of Algorithm 1. The first two were already suggested in [28], while the decomposition algorithm and valid inequalities are specific to $kRNL$ studied herein.

Nested partial partitioning In our problem, there are $|E|$ capacity constraints leading to a total of as many as $|E|$ active uncertain parameters and, so, $|E|$ subsets associated to the repartition of each subset D_k of the previous partition. Using this approach without modifications, at each iteration n of our method, there will be as many as $|E|^{(n-1)}$ subsets of a partition. This will possibly impact time performance due to the increase in the number of variables and constraints.

In order to mitigate the effect above, the authors in [28] suggest two practical approaches as trade-offs between computation time and quality of solution. One is to only take a subset of active samples from generated constraints, the ones with lowest slack, since constraints with greater slack are less likely to constrain the solution even with additional iterations. The other is based on the fact that the value of the objective is given by the worst result if we consider each element (D_k) of a partition. Only the elements (D_k) of the partition with this worst objective value restrict the overall objective and are called active subsets of a partition. If a subset is not active, further partitioning it is less likely to improve the objective. Applying these approaches will reduce the number of subsets of the partition at the next iteration, improving computational efficiency. The overall number of subsets defined at each iteration will then be algorithmic dependent.

Bounds Given a lower bound on the dynamic solution, we can use it to estimate the gap between our k-adaptive solution and the dynamic solution. Specifically, the bound is obtained by solving the problem with dynamic routing for the finite uncertainty set T, which contains all active uncertain parameters generated through different iterations presented in Section 3.4.

Benders decomposition For bifurcated routings, we use a Benders branch-and-cut approach and break the problem into one that designs edge capacities (master

problem defining variables $x_{ij}, ij \in E$) and a sequence of other problems that checks feasibility of the designed edge capacities with respect to the uncertain demand requirements. This decomposition of the problem is not applicable to non-bifurcated routings since routing templates variables are integer.

Specifically, given an integer solution $\tilde{x}_{ij}$ for the master problem, typically obtained as a feasible solution of a node in the branch-and-bound tree algorithm, we solve a feasibility subproblem for each subset of the partition. Notice that each subproblem is the following robust linear program

$$\begin{aligned}
\min \quad & \sum_{ij \in E} s_{ij} \\
\text{s.t.} \quad & \sum_{q \in Q} \sum_{p \in P(q)} d^q y^k_{qp} \delta^p_{ij} - s_{ij} \leq \tilde{x}_{ij} && ij \in E, \forall d \in D_k \\
& \sum_{p \in P(q)} y^k_{qp} = 1 && q \in Q \\
& s_{ij} \geq 0, y^k_{qp} \geq 0 && q \in Q, p \in P(q), ij \in E,
\end{aligned}$$

where variables s_{ij} are additional slack variables. Each subproblem is defined by a subset D_k of D, given by its matrix definition $A_k^{M' \text{ x } |Q|} d \leq b_k^{M'}$. Each subset D_k is a a polyhedral uncertainty set defined by an active uncertain parameter. The first constraint can be rewritten as

$$\max_{d \in D_k} \sum_{q \in Q} \sum_{p \in P(q)} d^q y^k_{qp} \delta^p_{ij} - s_{ij} \leq \tilde{x}_{ij}, \quad ij \in E \tag{3.14}$$

Using a classical technique based on LP duality (e.g., [20]), we can replace the maximization in (3.14) by its dual, obtaining the following LP:

$$\begin{aligned}
\min \quad & \sum_{ij \in E} s_{ij} \\
\text{s.t.} \quad & \sum_{m \in \{1,\ldots,M'\}} b_k(m) u^m_{ij} \leq \tilde{x}_{ij} + s_{ij} && ij \in E && (3.15) \\
& \sum_{p \in P(q)} y^k_{qp} = 1 && q \in Q && (3.16) \\
& \sum_{m \in \{1,\ldots,M'\}} A^T_k(q,m) u^m_{ij} \geq \sum_{p \in P(q)} \delta^{ij}_{qp} y^k_{qp} && ij \in E, q \in Q \\
& u^m_{ij} \geq 0, y^k_{qp} \geq 0, s_{ij} \geq 0 && q \in Q, p \in P(q), \\
& && ij \in E, m \in \{1, \ldots, M'\}
\end{aligned}$$

$A^T_k(q,m)$ and $b_k(m)$ indicate elements of their respective matrices. Let $\pi_{ij} \leq$

$0, \forall ij \in E$ and $\lambda_q, \forall q \in Q$ be the dual variables of constraints (3.15) and (3.16), respectively, and let $\tilde{\pi}_{ij}$ and $\tilde{\lambda}_q$ denote the optimal dual solution. By linear programming strong duality the solution for the subproblem above can be given by $\sum_{ij \in E} \tilde{x}_{ij}\tilde{\pi}_{ij} + \sum_{q \in Q} \tilde{\lambda}_q$. Moreover, the value of its solution is equal to 0 if and only the feasible set is nonempty. Hence, if $\sum_{ij \in E} \tilde{x}_{ij}\tilde{\pi}_{ij} + \sum_{q \in Q} \tilde{\lambda}_q > 0$ we add a strengthened mixed integer rounding [48] Benders cut to the master problem, where $f_0 = -\sum_{q \in Q} \tilde{\lambda}_q - \lfloor -\sum_{q \in Q} \tilde{\lambda}_q \rfloor$ and $f_{ij} = \tilde{\pi}_{ij} - \lfloor \tilde{\pi}_{ij} \rfloor$ and

$$\sum_{ij \in E} (\lfloor \tilde{\pi}_{ij} \rfloor + \frac{\max(f_{ij} - f_0, 0)}{1 - f_0}) x_{ij} \leq \lfloor -\sum_{q \in Q} \tilde{\lambda}_q \rfloor$$

If the master solution $\tilde{x}$ is not violated by the strengthened Benders cut we add the non-rounded cut instead.

We take advantage of nonzero values solutions of the slack variables (s_{ij}) and implement a heuristic solution for the master problem. For each subproblem, nonzero value solutions of the slack variables correspond to the capacity value missing to transform the master problem solution into a feasible one. For each solution $\tilde{x}_{ij}$ we select the maximum nonzero slack value provided by the different subproblems. These nonzero values of slack variables complement master problem solutions and, rounded up, are used as upper bounds for the master problem.

Cut-set inequalities Consider a partition of the node set V given by sets S_1 and $\bar{S}_1$, and let $E(S_1, \bar{S}_1)$ and $Q(S_1, \bar{S}_1)$ be the set of edges and commodities with extremities in different sets of the partition. The cut-set inequality associated with the partition states that the amount of capacity installed on edges in $E(S_1, \bar{S}_1)$ should be not less than the rounded up sum of the demands of commodities in $Q(S_1, \bar{S}_1)$. We have our cut-set inequality given by

$$\sum_{ij \in E(S_1, \bar{S}_1)} x_{ij} \geq \lceil \max_{d \in D} \sum_{q \in Q(S_1, \bar{S}_1)} d^q \rceil$$

Since x is a first stage decision variable cut-set inequalities have to hold for the original uncertainty set.

We separate robust cut-set inequalities using two steps: 1) the master problem starts with cuts generated for each node of the network and at each integer solution, and 2) during branching, we separate cuts heuristically by generating a two-subset random partition of the nodes and then performing a local search picking up one node and moving it to the other subset until there is no more improvement in the violation. If no violated inequality is found we repeat the procedure up to a maximum of 20 iterations.

3.6 Implementation and results

We first present a summary indicating the purpose and main achievements of our experiment, followed by details of our implementation and the results themselves.

3.6.1 Summary of results

Purpose The purpose of the computational experiments is two-fold:

- We assess the numerical complexity of the k-partitioning scheme, including the effect of the valid inequalities and Benders decomposition algorithm.
- We assess the cost reduction provided by that scheme over static routing.

Main achievements We were able to show that the k-adaptive partitioning scheme provides, in general, less conservative solutions than previous algorithms presented in the literature. We also show that the decomposition algorithms developed here were able to improve time performance for the more complex instances.

3.6.2 Instances and implementation

Instances Six networks available from SNDlib [91] and eight networks randomly generated were utilized with the characteristics given in Table 3.1. The first six networks presented are as referenced in [84]. The other networks are randomly generated and inspired by the instances used in [98] so that each node has at least two incoming edges to avoid trivial problems. Edge costs are randomly generated integer numbers in $\{1, \ldots, 5\}$.

Each commodity $q \in Q$ is associated with the uncertain demand d^q, within a given uncertainty set based in [31]. The demand for commodity $q \in Q$ varies around a given nominal demand $\bar{d}^q \geq 0$ with a maximal possible deviation of $\hat{d}^q \geq 0$, that is, $d^q(\xi) = \bar{d}^q + \xi^q \hat{d}^q$ for $\xi \in \Xi$ where $\Xi = \{\xi \mid \sum_{q \in Q} \xi^q \leq \Gamma,\ 0 \leq \xi^q \leq 1\}$. Demands $\bar{d}^q$ are randomly generated integer numbers in $\{1, \ldots, 100\}$ and maximal deviations $\hat{d}^q$ are randomly generated numbers in $\{0.1\bar{d}^q, \ldots, 0.7\bar{d}^q\}$. We choose the value of Γ according to the probabilistic bound introduced by [31]. Namely, we set two levels of guaranteed probabilistic bound (denoted ϵ : 0.25, 0.01). Then, for each value of ϵ, the bound from [31] prescribes a value Γ^ϵ such that all feasible solutions for the problem satisfy the following property: if demands for each commodity $q \in Q$ are symmetric and independent random variables distributed in $[\bar{d}_q - \hat{d}_q, \bar{d}_q + \hat{d}_q]$, then, for each $ij \in E$, the probability that the flow exceeds the capacity installed on edge ij is less than ϵ.

We test these networks, with the two different Γs, giving a total of 28 instances tested.

For the abilene network a test case with all possible Γs was also utilized. We have predetermined a maximum of 5 paths for each commodity (if they exist). These paths are calculated as the 5 shortest edge cost paths.

Routing schemes We experiment with many of the different routing schemes presented in this study, according to the configurations in Table 3.2. We note the following:

- We further split the routing schemes into routing schemes with bifurcated (BF) and non-bifurcated (NBF) flows, when applicable.
- $\mathbb{F}^{stat}$ refers to static routing, where for each commodity the same paths and splitting between paths are used independent of the realization of demand.
- $\mathbb{F}^{VS}$ refers to volume routing as defined in Section 3.2.2.
- $\mathbb{F}^{2|}$ refers to 2 partitioning routing also as defined in Section 3.2.2.
- $\mathbb{F}^{k}St$ is k-adaptive partition where the solution for each subset of the partition scheme is static, as was presented in this study.
- $\mathbb{F}^{k}Vl$ is a variant of k-adaptive partition where we adopt volume routing solutions for each subset of the partition.
- We also split the k-adaptive partition routing schemes, $\mathbb{F}^{k}St$ and $\mathbb{F}^{k}Vl$, into Full Partitioning (FP), meaning that active uncertain parameters related partitions are created for all capacity constraints (3.10), and Partial Partitioning (PP), that reflects the nested partial partition described in Section 3.5.
- For the reasons already mentioned and experimented in [84], due to large formulations that become hard to solve, we have suppressed affine routings from our tested routing schemes.

Algorithms Specification Algorithms were coded in Julia [79] using JuMP, JuMPeR and BlockDecomposition packages and Cplex 12.7. All algorithms were run in an Intel CORE i7 CPU 3770 machine. A limit of 7200 seconds of computing time was given for each instance and iteration.

Algorithms are listed in Table 3.3 for bifurcated flow routings and Table 3.4 for non-bifurcated flow routings:

- Each algorithm listed was adapted to run with different routing schemes, as indicated in columns 3 to 5 of Table 3.3 and Table 3.4. Additionally, except for the 2-partition algorithm (TP), they can run with or without addition of cut-set inequalities.

- The Compact formulation algorithm was implemented using a classical robust dual reformulation to solve the original formulation of the problem, according to each routing scheme.
- The CPLEX BENDERS algorithm was implemented applying the new automatic Benders decomposition functionality of Cplex 12.7.
- Root node only cut-set inequalities (RC) were introduced to the CPLEX BENDERS Algorithm because Cplex 12.7 does not support running lazy callback functions together with its Benders Decomposition feature.
- The Adapted Benders Decomposition algorithm implements the Benders method presented in this study.
- The 2-partition algorithms were implemented replicating the algorithm presented in [129], extended to integer capacities.

We have applied decomposition algorithms only for Static Partial Partitioning routing as an experiment to verify their time performance compared to Compact reformulations.

3.6.3 Results

Solution costs Tables 3.5 and 3.6 present solution improvement over static routing provided by different bifurcated and non-bifurcated routing schemes with two different Γs. For Table 3.5 there are a number of cases that do not achieve an optimal solution under the time limit and they appear marked as M. For Table 3.6 instances missing correspond to cases where there was no improvement in solution or to cases that did not achieve an optimal solution under the time limit among all routings. Solution improvement is given as percentage by the formula $100\% * \frac{(Static\,Solution - Solution)}{Static\,Solution}$. For partitioning routing schemes, different algorithms can give different solutions because of multiple possible partitioning alternatives between iterations. We present here the best result among algorithms for these cases. The columns referenced as Bound correspond to the solution of the Bound procedure at the last iteration of our nested iterative k-adaptive algorithm and provides a reference of how much space for reducing conservatism is still available.

Figure 3.2 presents solution improvement in percentage over static routing provided by different bifurcated routing schemes, for the abilene network and $0 \leq \Gamma \leq |Q|$. Routings $\mathbb{F}^k VlPP$ and $\mathbb{F}^k VlFP$ are not presented since they did not improve over $\mathbb{F}^{VS}$ in this case.

Table 3.9 summarizes the analysis we make of Tables 3.5 and 3.6 regarding the performance of each routing scheme in improving the static result. This analysis indicates:

- There is a short gap between static and dynamic solutions. Nevertheless, the routing schemes implemented were able to reduce this gap in many cases.
- Cost reduction is higher for lower Γs. From theory we know already that increasing Γ leads to more conservatism, but we see also that all routing schemes have more difficulty in providing better solutions for higher Γs.
- The analysis of routings based on partitioning shows that $\mathbb{F}^{2|}$ provided the least solution improvement. In part this was due to the time limit and because the partition hyperplane direction was arbitrarily chosen. In that sense, $\mathbb{F}^k PP$ and $\mathbb{F}^k FP$ had more success, for Static and especially for Volume solutions. In particular, $\mathbb{F}^k PP$ for Static and Volume solutions was able to give results comparable to $\mathbb{F}^k FP$ and in some instances even better.
- Volume solutions, both for $\mathbb{F}^{VS}$ and $\mathbb{F}^k$, were able to give the best improvements, but it cannot handle non-bifurcated flows. $\mathbb{F}^k Vl$ was able to further improve $\mathbb{F}^{VS}$.
- For non-bifurcated flows many instances were not solved within the time limit. Moreover, solution improvement was not as large as in the bifurcated case. This suggests we need to consider more path flow options for the non-bifurcated case, and also develop more efficient decomposition algorithms.

Computation Times Table 3.7 presents the total time for different algorithms running for bifurcated routing schemes. We have limited our algorithms to 2 iterations for full partitioning and 3 iterations for partial partitioning. There are a number of instances that do not achieve an optimal solution under the time limit, marked as M. Table 3.8 presents a comparison between the two different decomposition algorithms we have used, CPLEX BENDERS and Adapted Benders Decomposition. They were run for static bifurcated partial partition routings. Here we present total time (TT) and also number of Benders cuts (BC) and cut-set cuts (CSC) introduced by each algorithm. We reproduce total time (TT) columns for the compact formulation with cut-set and without cut-set inequalities added to facilitate comparison between decomposition and non-decomposition algorithms. Figures 3.3 and 3.4 compare algorithms for static bifurcated partial partition routing through a performance profile [58]. In these figures the vertical axis points out in percentage, for each algorithm, in how many instances the result was not more than x times - horizontal axis - worse than the best algorithm. These figures present a separate analysis for

real life, more complex, SNDlib instances and randomly generated instances. Using the first plot, $x = 1$, we can see that the $DBD+C$ algorithm was best in 50% of the cases for SNDlib instances and the C algorithm was best in approximately 58% of the cases for random instances. Table 3.10 summarizes the analysis we make of Tables 3.7 and 3.8 regarding time performance of each algorithm and routing scheme. We remark that cut-set inequalities are randomly generated and can influence the results of comparison. We are interested in verifying the effectiveness of introducing cut-set inequalities to our formulations, the use of decomposition algorithms, the new Benders functionality of CPLEX 12.7, the use of partial partitioning and volume versus static partitioning.

Table 3.10 and further analysis of Tables 3.7, 3.8 and Figures 3.3 and 3.4 indicate:

- Robust cut-sets inequalities, in general, were key to reduce branching and providing better time performance.

- Figure 3.3 shows that, for more complex SNDlib networks, Benders based decomposition algorithms presented better time performance when compared to compact formulations algorithms. It shows also that our Adapted Benders Decomposition implementation was in general better in performance than CPLEX 12.7 Benders Decomposition. This is due mainly to the use of robust cut-set inequalities. On the contrary, by Figure 3.4 and for randomly generated networks, the compact formulation algorithms presented overall better time performance.

- For Adapted Benders Decomposition the introduction of robust cut-sets inequalities resulted in a significant reduction in the number of Benders cuts.

- Although $\mathbb{F}^kPP$ takes more iterations than $\mathbb{F}^kFP$, it requires less time. In that sense, and the fact that it delivers comparable solutions, $\mathbb{F}^kPP$ presented itself as a good option to $\mathbb{F}^kFP$.

- The analyses of routes based on partitioning shows that $\mathbb{F}^{2|}$ provided the worst time performance, due to the associated non-polynomial binary search algorithm. Based on that and on solution improvement, $\mathbb{F}^kPP$ and $\mathbb{F}^kFP$ turned out to be preferred partitioning based routing schemes.

- As identified above, many non-bifurcated flow instances were not resolved within the time limit, suggesting the need to develop more efficient decomposition algorithms.

- In general, for our instances, $\mathbb{F}^kVl$ has worse time performance than $\mathbb{F}^kSt$, but provides better bifurcated solutions as we have seen.

Network	| V |	| E|	| Q |	$\Gamma^{0.25}$	$\Gamma^{0.01}$
abilene	12	15	65	6	19
polska	12	18	66	6	19
pdh	11	34	24	4	12
di-yuan	11	42	22	4	11
nobel-us	14	21	91	7	23
atlanta	15	22	105	7	24
n10e14c14-1	10	14	14	3	10
n10e14c14-2	10	14	14	3	10
n10e14c19-1	10	14	19	4	11
n10e14c19-2	10	14	19	4	11
n10e19c14-1	10	19	14	3	10
n10e19c14-2	10	19	14	3	10
n10e19c19-1	10	19	19	4	11
n10e19c19-2	10	19	19	4	11

Table 3.1: Network profiles

Code	Routing Scheme
$\mathbb{F}^{stat}BF$	Static BF
$\mathbb{F}^{stat}NBF$	Static NBF
$\mathbb{F}^{VS}BF$	Volume BF
$\mathbb{F}^{2\|}BF$	Two Partitioning BF
$\mathbb{F}^{k}StPPBF$	Static Partial Partitioning BF
$\mathbb{F}^{k}StPPNBF$	Static Partial Partitioning NBF
$\mathbb{F}^{k}VlPPBF$	Volume Partial Partitioning BF
$\mathbb{F}^{k}StFPBF$	Static Full Partitioning BF
$\mathbb{F}^{k}StFPNBF$	Static Full Partitioning NBF
$\mathbb{F}^{k}VlFPBF$	Volume Full Partitioning BF

Table 3.2: Routing schemes

Code	Algorithm	$\mathbb{F}^{stat}BF$ $\mathbb{F}^{k}StPPBF$	$\mathbb{F}^{VS}BF$ $\mathbb{F}^{k}VlPPBF$ $\mathbb{F}^{k}VlFPBF$ $\mathbb{F}^{k}StFPBF$	$\mathbb{F}^{2\|}BF$
C	Compact reformulation	√	√	-
C+C	Compact reformulation + cut-set	√	√	-
DCP	CPLEX BENDERS	√	-	-
DCP+RC	CPLEX BENDERS + root cut-set	√	-	-
DBD	Adapted Benders Decomposition	√	-	-
DBD+C	Adapted Benders Decomposition + cut-set	√	-	-
TP	2-partition	-	-	√

Table 3.3: BF algorithms implemented

Code	Algorithm	$\mathbb{F}^{stat}NBF$	$\mathbb{F}^{k}PPNBF$	$\mathbb{F}^{k}StFPNBF$
C	Compact reformulation	√	√	√
C+C	Compact reformulation + cut-set	√	√	√

Table 3.4: NBF algorithms implemented

		Routing Scheme (all BF) Solution						
Network	Γ	$\mathbb{F}^{VS}$	$\mathbb{F}^{2\|}$	$\mathbb{F}^{k}StPP$	$\mathbb{F}^{k}StFP$	$\mathbb{F}^{k}VlPP$	$\mathbb{F}^{k}VlFP$	Bound
abilene	6	1.92	0.69	0.96	0.32	1.92	1.92	2.24
	19	0.27	0.27	0.27	0.27	0.27	0.27	3.26
pdh	4	9.20	M	2.68	3.83	10.34	M	16.09
	12	2.71	M	0.68	M	M	M	16.61
polska	6	0.12	0.12	0.23	0.12	1.06	1.53	14.20
	19	0.08	M	0.12	0.08	0.08	0.08	14.77
di-yuan	4	1.90	1.14	1.52	1.33	2.86	2.86	11.62
	11	0.10	0.10	0.10	0.10	0.10	0.10	8.19
nobel-us	7	8.14	M	2.37	4.75	9.49	M	16.61
	23	1.55	M	0.31	0.93	2.48	M	19.81
atlanta	7	4.00	M	1.00	2.00	4.50	4.50	11.00
	24	1.40	M	0.93	M	M	M	12.09
n10e14c14-1	3	1.89	0.21	1.08	0.97	2.55	2.32	6.03
	10	0.05	0	0.05	0.05	0.05	0.05	5.84
n10e14c14-2	3	1.86	1.18	0.24	0.65	2.15	1.91	3.01
	10	0	0	0	0	0	0	1.78
n10e14c19-1	4	4.67	0	0.21	2.20	4.92	4.67	9.19
	11	0	0	0	0	0	0	9.30
n10e14c19-2	4	2.61	2.45	0.87	2.39	3.58	3.60	9.62
	11	0	0	0	0	0	0	5.23
n10e19c14-1	3	2.85	0	0.91	1.66	4.03	3.58	7.16
	10	0	0	0	0	0	0	9.37
n10e19c14-2	3	1.02	0.38	0.18	0.51	1.74	1.36	7.39
	10	0	0	0	0	0	0	15.88
n10e19c19-1	4	1.87	0	1.21	1.74	2.94	2.63	5.25
	11	0	M	0	0	0	0	5.25
n10e19c19-2	4	2.97	M	1.06	1.54	4.63	4.28	14.74
	11	0	0	0	0	0	0	22.64

Table 3.5: Percentage of solution improvement over static BF routing

		Routing Scheme (all NBF) Solution		
Network	Γ	$\mathbb{F}^{k}StPP$	$\mathbb{F}^{k}StFP$	Bound
abilene	6	0.31	0.31	10.13
	19	0.08	0.08	5.59
di-yuan	4	0.19	0.19	13.17
	11	0.07	0.07	17.06
atlanta	7	0.07	0.07	10.03
	24	0.50	0.50	26.37
n10e14c19-1	4	0.11	0.11	12.33
n10e14c19-2	4	0.54	0.54	10.11

Table 3.6: Percentage of solution improvement over static NBF routing

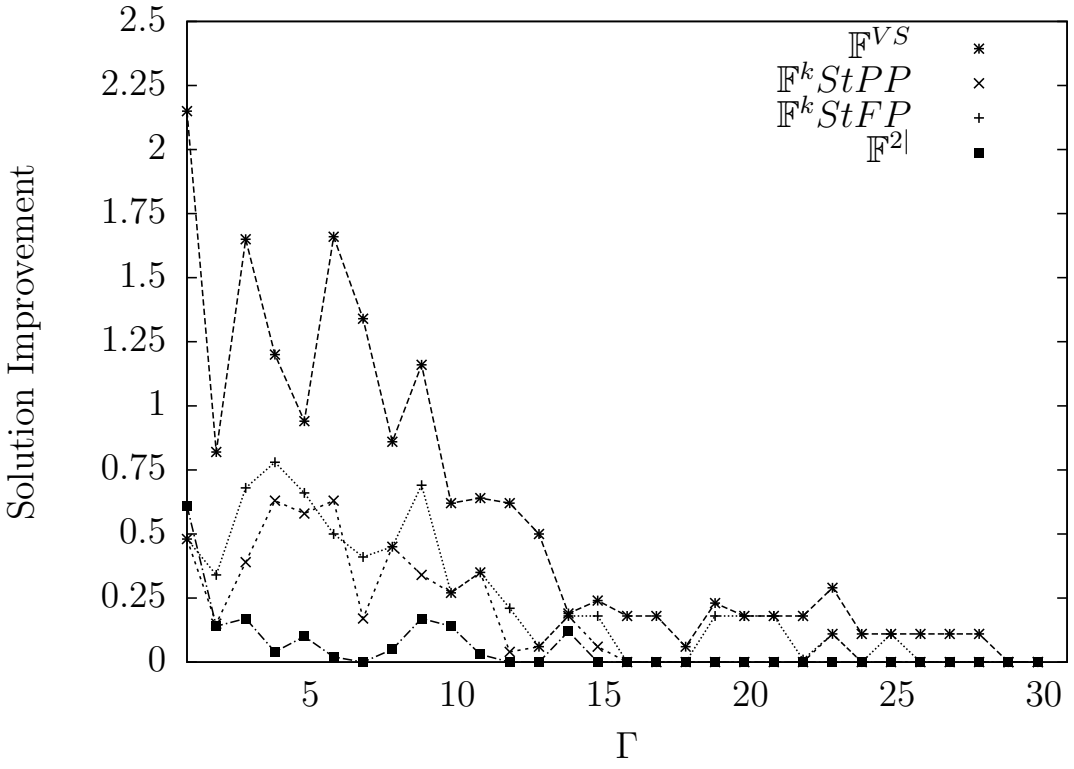

Figure 3.2: Percentage of solution improvement over static BF routing for the abilene network and different Γs

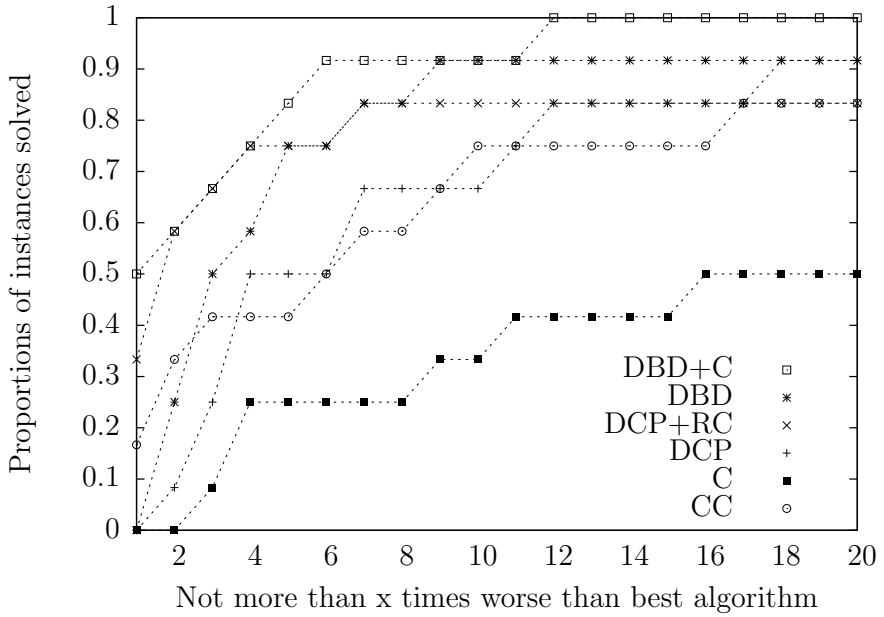

Figure 3.3: Performance profile of decomposition algorithms for SNDlib networks

		Routing Scheme (all BF) and Algorithm Code								
		$\mathbb{F}^{stat}$	$\mathbb{F}^{VS}$	$\mathbb{F}^{k}VlPP$		$\mathbb{F}^{k}VlFP$	$\mathbb{F}^{2\|}$	$\mathbb{F}^{k}StPP$		$\mathbb{F}^{k}StFP$
Network	Γ	C+C	C	C	C+C	C+C	C	C	C+C	C+C
abilene	6	2.20	2.52	3.15	2.14	1.62	71.44	11.03	2.84	4.29
	19	0.55	1.55	543.30	241.79	473.81	3676.84	352.42	53.25	162.87
polska	6	2.52	8.72	4448.67	2761.89	M	M	919.74	409.78	1341.70
	19	5.82	7.77	M	M	M	M	M	M	M
pdh	4	5.24	11.89	4728.78	3815.63	292.49	6534.29	M	5572.47	17.88
	12	11.55	11.15	M	1742.02	1379.20	M	M	3310.10	4336.65
di-yuan	4	0.87	2.42	1648.51	280.80	572.36	6943.07	807.35	152.53	157.43
	11	0.09	0.24	69.53	5.82	1.70	7123.09	101.17	9.86	3.81
nobel-us	7	2.72	1.22	7406.52	2252.20	M	M	1006.32	455.20	425.03
	23	23.09	6.72	M	8838.02	M	M	M	4916.07	5213.05
atlanta	7	1.85	3.88	1688.62	1531.52	M	M	1625.36	1196.10	2034.70
	24	8.11	14.95	M	M	M	M	M	2979.89	M
n10e14c14-1	3	0.01	0.01	0.82	0.93	0.15	66.74	0.88	1.20	2.04
	10	0.01	0.01	0.61	0.58	0.62	5.83	0.06	0.07	0.15
n10e14c14-2	3	0.02	0.01	0.83	0.75	0.15	16.19	0.63	0.58	0.42
	10	0.01	0.01	0.36	0.41	0.53	4.13	0.05	0.06	0.10
n10e14c19-1	4	0.02	0.02	1.26	1.15	1.12	25.69	20.62	0.31	44.62
	11	0.01	0.01	1.08	1.18	0.08	776.07	0.22	0.25	0.27
n10e14c19-2	4	0.01	0.01	4.20	2.45	0.74	69.82	1.45	0.91	0.65
	11	0.01	0.01	0.67	0.67	0.74	8.18	0.12	0.14	0.10
n10e19c14-1	3	0.02	0.01	1.78	0.32	0.22	2320.30	11.71	6.40	8.20
	10	0.01	0.01	0.51	0.66	0.86	42.22	0.17	0.17	0.16
n10e19c14-2	3	0.03	0.01	2.15	1.76	1.61	222.64	0.89	0.30	1.77
	10	0.01	0.01	0.42	0.49	0.06	11.23	0.16	0.15	0.12
n10e19c19-1	4	0.02	0.02	66.10	21.67	0.44	4131.30	87.89	10.44	66.40
	11	0.01	0.01	1.01	0.89	1.10	M	0.28	0.25	0.28
n10e19c19-2	4	0.01	0.01	6.51	2.78	0.51	M	2.64	31.60	12.15
	11	0.01	0.01	1.31	0.73	0.88	13.42	0.33	0.32	0.36

TT: Total time in seconds
BC: Number of Benders cuts
CSC: Number of Cut-set cuts

Table 3.7: Bifurcated flows total time performance (s)

		Algorithms for $\mathbb{F}^k StPPBF$													
		C	C+C	DCP			DCP+RC			DBD			DBD+C		
Network	Γ	TT	TT	TT	BC	CSC	TT	BC	CSC	TT	BC	CSC	TT	BC	CSC
abilene	6	11.03	2.84	49.38	69	0	199.46	673	24	10.69	549	0	5.23	255	54
	19	352.42	53.25	410.09	140	0	M	M	M	27.03	758	0	10.32	327	47
polska	6	919.74	409.78	461.35	1124	0	261.54	2791	24	625.33	4435	0	1174.26	5536	53
	19	M	M	1597.32	501	0	976.20	918	24	590.18	2476	0	509.96	1551	63
pdh	4	M	5572.47	2184.11	3294	0	339.46	1571	22	9496.00	9168	0	3941.11	6337	57
	12	M	3310.10	265.93	1122	0	101.34	686	22	210.46	5582	0	92.45	2243	57
di-yuan	4	807.35	152.53	125.65	78	0	52.86	169	22	169.33	3740	0	199.10	2487	74
	11	101.17	9.86	66.36	18	0	20.83	14	22	68.00	2934	0	23.77	1936	66
nobel-us	7	1006.32	455.20	1496.64	4982	0	389.74	2210	28	3291.28	8534	0	2001.89	6997	58
	23	M	4916.07	5911.77	3710	0	1795.92	2805	28	2813.23	6384	0	565.42	3442	56
atlanta	7	1625.36	1196.10	665.17	439	0	309.20	584	30	302.12	3307	0	195.95	2260	30
	24	M	2979.89	3654.80	296	0	2052.01	168	30	387.00	3359	0	328.13	2957	28
n10e14c14-1	3	0.88	1.20	1.10	109	0	1.9	184	20	1.73	608	0	1.50	463	45
	10	0.06	0.10	0.34	122	0	0.34	98	20	1.20	355	0	1.20	170	44
n10e14c14-2	3	0.68	0.80	1.0	10	0	0.77	60	20	5.52	500	0	6.17	236	45
	10	0.05	0.06	0.32	81	0	0.30	74	20	1.4	414	0	0.86	172	47
n10e14c19-1	4	0.62	0.31	7.2	148	0	84.4	40	20	5.41	716	0	7.80	700	50
	11	0.22	0.25	0.86	179	0	0.9	138	20	1.30	295	0	1.23	252	52
n10e14c19-2	4	1.45	0.91	2.70	144	0	2.3	209	20	2.51	728	0	2.30	583	71
	11	0.12	0.14	0.46	77	0	0.39	97	20	0.72	308	0	0.53	99	105
n10e19c14-1	3	17.71	6.40	7.30	364	0	3.30	183	18	6.30	1461	0	2.71	1013	47
	10	0.17	0.17	0.64	145	0	0.56	142	18	0.85	587	0	0.43	303	47
n10e19c14-2	3	0.89	0.30	3.70	51	0	2.95	46	20	3.72	491	0	2.53	386	41
	10	0.16	0.15	0.47	143	0	0.54	108	20	0.73	444	0	0.73	169	78
n10e19c19-1	4	87.89	10.44	6.70	90	0	29.3	77	20	4.40	1353	0	3.52	612	79
	11	0.28	0.29	1.00	156	0	1.10	85	20	2.30	1835	0	1.31	304	49
n10e19c19-2	3	2.64	31.60	42.00	110	0	13.7	129	20	10.74	1000	0	8.40	581	47
	10	0.33	0.32	1.53	206	0	1.10	110	20	1.85	1235	0	0.93	587	62

TT: Total time in seconds
BC: Number of Benders cuts
CSC: Number of Cut-set cuts

Table 3.8: Algorithms time performance and number of cuts for partial partitioning and bifurcated static routings

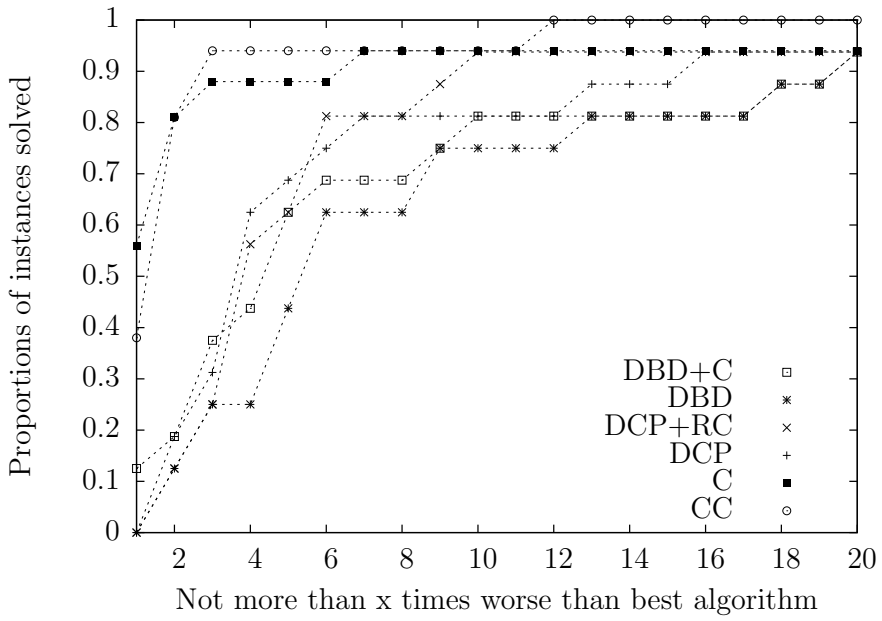

Figure 3.4: Performance profile of decomposition algorithms for randomly generated networks

Routing		Number of Cases of Solution Improvement comparing to Static		
Split	Scheme	$\Gamma^{0.25}$	$\Gamma^{0.01}$	Total
BF	$\mathbb{F}^{VS}$	14 of 14	7 of 14	21 of 28
	$\mathbb{F}^{2\|}$	7 of 14	2 of 14	9 of 28
	$\mathbb{F}^{k}StPP$	14 of 14	7 of 14	21 of 28
	$\mathbb{F}^{k}StFP$	14 of 14	5 of 14	19 of 28
	$\mathbb{F}^{k}VlPP$	14 of 14	5 of 14	19 of 28
	$\mathbb{F}^{k}VlFP$	12 of 14	4 of 6	16 of 28
NBF	$\mathbb{F}^{k}StPP$	5 of 14	3 of 14	8 of 28
	$\mathbb{F}^{k}StFP$	5 of 14	3 of 14	8 of 28

Table 3.9: Summary for solution improvement comparison

Comparison	Cases compared*	Number of better Time Performance	
		SNDlib instances	random instances
Cut-set versus Non cut-set	$\mathbb{F}^{k}StPP$ C+C $\leq$ $\mathbb{F}^{k}StPP$ C	12 of 12	7 of 16
	$\mathbb{F}^{k}StPP$ DBD+C $\leq$ $\mathbb{F}^{k}StPP$ DBD	10 of 12	13 of 16
	$\mathbb{F}^{k}StPP$ DCP+RC $\leq$ $\mathbb{F}^{k}StPP$ DCP	10 of 12	10 of 16
	$\mathbb{F}^{k}VlPP$ C+C $\leq$ $\mathbb{F}^{k}VlPP$ C	12 of 12	12 of 16
Decomposition versus Non Decomposition	$\mathbb{F}^{k}StPP$ DBD+C $\leq$ $\mathbb{F}^{k}StPP$ C+C	7 of 12	3 of 16
	$\mathbb{F}^{k}StPP$ DBD $\leq$ $\mathbb{F}^{k}StPP$ C	10 of 12	3 of 16
	$\mathbb{F}^{k}StPP$ DCP+RC $\leq$ $\mathbb{F}^{k}StPP$ C+C	9 of 12	2 of 16
	$\mathbb{F}^{k}StPP$ DCP $\leq$ $\mathbb{F}^{k}StPP$ C	9 of 12	2 of 16
Adapted Benders Decomposition versus CPLEX BENDERS	$\mathbb{F}^{k}StPP$ DBD+C $\leq$ $\mathbb{F}^{k}StPP$ DCP	8 of 12	7 of 16
	$\mathbb{F}^{k}StPP$ DBD+C $\leq$ $\mathbb{F}^{k}StPP$ DCP+RC	6 of 12	9 of 16
	$\mathbb{F}^{k}StPP$ DBD $\leq$ $\mathbb{F}^{k}StPP$ DCP	7 of 12	5 of 16
Partial versus Full Partitioning	$\mathbb{F}^{k}StPP$ C+C$\leq$ $\mathbb{F}^{k}StFP$ C +C	9 of 12	10 of 16
	$\mathbb{F}^{k}VlPP$ C+C$\leq$ $\mathbb{F}^{k}VlFP$ C +C	8 of 12	7 of 16
Static Partitioning versus Volume Partitioning	$\mathbb{F}^{k}StFP$ C+C$\leq$ $\mathbb{F}^{k}VlFP$ C+C	9 of 12	7 of 16
	$\mathbb{F}^{k}StPP$ C+C$\leq$ $\mathbb{F}^{k}VlPP$ C +C	8 of 12	13 of 16

$^{*} \leq$ *indicates better than or equal*

Table 3.10: Summary for bifurcated flows time performance comparison

Chapter 4

Exact Solution Algorithms for Minimizing the Total Tardiness with Processing Time Uncertainty

4.1 Introduction

Scheduling is an important and diverse topic in combinatorial optimization. It has applications in many different industries, ranging from production and manufacturing systems to transportation and logistics systems. The scheduling problem considered in this study considers a unique machine. Hence, we are given a set of n jobs, denoted $N = \{1, 2, \ldots, n\}$, and any feasible solution for the problem consists of a permutation of N, represented by σ where $\sigma(k)$ denotes the job that occupies position k for each $k \in N$. The objective function considered herein supposes that each job $i \in N$ has a given due date d_i and a processing time p_i. Assuming that the processing times are known with precision, we can define the completion time of job i for σ as

$$C_i(\sigma) = \sum_{k=1}^{\sigma^{-1}(i)} p_{\sigma(k)},$$

and the tardiness of job i can be defined as $T_i(\sigma) = \max(C_i(\sigma) - d_i, 0)$. Denoting by X the set of all permutations of N, the problem that minimizes the tardiness can be formally stated as

$$\min_{\sigma \in X} \sum_{i=1}^{n} T_i(\sigma). \tag{4.1}$$

Notice that problem (4.1) is often denoted as $1||\sum_j T_j$ in the literature, where 1 means that a single machine is used and $\sum_j T_j$ represents the objective function involved.

In real applications, the processing times are hardly known with precision (see

the examples below). Hence, it is more realistic to assume that p can take any value within a given set U that contains plausible values for p. Then, the robust counterpart of (4.1) is

$$\min_{\sigma \in X} \max_{p \in U} \sum_{i=1}^{n} T_i(p, \sigma), \tag{4.2}$$

where we introduced p in the definition of T_i to emphasize that its value depends on $p \in U$. In this study, we let U be the budgeted uncertainty set proposed by [31]. Hence, we consider a parameter Γ and, for each job i, a mean value $\bar{p}_i$ and a deviation $\hat{p}_i$. Then, the uncertainty set is defined as

$$U \equiv \left\{ p : p_i = \bar{p}_i + \hat{p}_i \xi_i,\ i = 1, \ldots, n; 0 \leq \xi_i \leq 1, \sum_{i=1}^{n} \xi_i \leq \Gamma \right\}, \tag{4.3}$$

where downwards deviations are disregarded because they are not used by worst-case scenarios. The motivation behind definition U is two-fold. First, the set is easy to characterize and build from historical data, as only lower and upper bounds on p are required, while the value of Γ let us model the risk-averseness of the decision maker. Specifically, higher values of Γ lead to larger uncertainty sets and thus, more conservative solutions. For instance $\Gamma = 0$ means that $U = \{\bar{p}\}$ while $\Gamma = n$ means that U is the box $[\bar{p}, \bar{p} + \hat{p}]$. Second, the set has a nice combinatorial structure that can be exploited by combinatorial algorithms to provide more efficient solution algorithms than using arbitrary uncertainty sets. In fact, these reasons have made (4.3) extremely popular in the robust combinatorial optimization and integer programming literature. In particular, recent papers on robust scheduling have provided polynomial algorithms for robust problems that would have been NP-hard using arbitrary uncertainty sets [37, 118].

The effect of uncertainty on scheduling has been studied under many different perspectives. For a more recent survey on different perspectives to scheduling under uncertainty one can see, for example, [40]. Applications of robust approaches to the total tardiness problem have been implemented in different industries. Typically, these applications solve problems where there are penalties associated with not fulfilling due dates and the decision maker has a conservative attitude in trying to minimize these penalties no matter the realization of uncertainty.

In [47] the authors study project scheduling at a large IT services delivery center in which there are unpredictable delays, but known to belong to a bounded set. They apply robust optimization to minimize tardiness while informing the customer of a reasonable worst-case completion time. In [87] the authors consider a scheduling problem in which manufacturing companies with large energy demand must comply with total energy consumption limits in specified time intervals and have to deal

with the fact that in reality the production schedules are not executed exactly as planned due to unexpected disturbances such as machine breakdowns or material unavailability. The goal is to find a robust schedule which minimizes total tardiness and guarantees that the energy consumption limits are not violated if the start times of operations are arbitrary delayed within a given limit. A robust total weighted tardiness approach is implemented in [4] for operation room planning in a hospital with uncertain surgery duration. The problem aims at minimizing a measure of waiting time of the patients and a tardiness function is created weighted by a patient urgency parameter.

In this work, we shall develop two types of exact solution algorithms to solve (4.2). The first type of algorithms combines the integer programming formulations available for $1||\sum_j T_j$ with the decomposition algorithm in the line of [128] and [8]. The second type of algorithms takes the (combinatorial) branch-and-bound algorithms developed for $1||\sum_j T_j$ and extend them to the robust counterpart (4.2).

Most exact solution algorithms for $1||\sum_j T_j$ are strongly based on dominance conditions and decomposition principles applied to the sequencing of jobs. These conditions and principles define ordering and positioning restrictions for the jobs in an optimal sequence and are used as a base to develop branch-and-bound and dynamic programming algorithms. For branch-and-bound algorithms different lower bound propositions were developed, including even the absence of lower bounds and relying only on the power of decomposition (for examples see [54, 73, 76, 101, 117]).

Decomposition principles were initially developed in [78] and establish conditions by which a single machine scheduling problem can be decomposed into subproblems that can be solved independently. For the robust scheduling case, due to the added complexity of having to deal with correlations of uncertainty between subproblems, we do not consider the decomposition principles introduced there.

Many integer formulation approaches can also be found in the literature for deterministic single machine problems in general (see [74] for review and comparisons). The formulations are based on the type of variables used to define the sequencing of jobs. Some effort has been made in order to improve performance of this approach by defining strong valid inequalities through polyhedral studies (see [103] for example).

Six different integer programming formulations of the single machine total tardiness problem are compared in [9]. They verify that a generic integer programming approach does not compete with state-of-the-art branch-and-bound tardiness algorithms. They conclude that the sequence-position formulation provides most computationally effective solutions and note that, although the attention paid to time-index formulations may be justified as they provide insight into theoretical results, they leave something to be desired when it comes to computational experience.

We review next some recent works on robust scheduling, which originated in the seminal paper from [50]. That seminal work has been followed by a sequence of works on the theoretical aspects of robust scheduling [e.g. 6, 82, 125] where the authors have shown that very simple scheduling problems become NP-hard as soon as the uncertainty set contains more than one scenario. Some authors have also studied the complexity of simple robust scheduling problems under budgeted uncertainty, see [37, 118]. We also refer to [119] for a recent survey on robust scheduling.

From a numerical viewpoint, the closest work from the current manuscript has been carried out in [55] where the authors study the optimal allocation of surgery blocks to operating rooms. Therein the authors introduce different models, including a robust model that is similar to the one studied here. However, a fundamental difference is that we let T_i be different for each $p \in U$, which is underlined by the notation $T_i(p, \sigma)$ used in (4.2), while [55] consider a static-model where T_i must be fixed independently from $p \in U$. While the static model is easier to handle computationally, it is also more conservative, as we shall illustrate briefly in our numerical experiments.

Our main contributions with respect to the literature are summarized below.

- We extend dominance rules developed for deterministic single machine total tardiness problem to our robust version.
- We develop a branch-and-bound algorithm based on the dominance rules above and a defined lower bound.
- We introduce different MILP formulations for our problem, and define dynamic programming routines to solve separation problems within a given decomposition algorithm.
- We compare the solution performance of our different MILP formulations and branch-and-bound algorithms.

Outline We formally present the problem with MILP formulations in Section 4.2. There, we also present the algorithm utilized to solve it. Next, in Section 4.3, we present the branch-and-bound algorithm developed to solve our problem. Sections 4.4 and 4.5 detail key concepts used in the algorithms presented. In Section 4.6 we present computational results.

4.2 MILP formulations

In [9] the authors summarize different MILP formulations for the deterministic single machine total tardiness problem. Based on the variables defined and their determin-

istic formulations, we present here the equivalent robust counterpart formulations utilized in our experiment.

The formulations are based on the type of variables defined to represent the sequencing of jobs (see [9]). We have utilized sequence-positions variables and linear ordering variables.

Sequence position variables are variables x such that $x_{ik} = 1$ if job i is in position k and 0 otherwise. Linear ordering variables are variables y such that $y_{ij} = 1$ if job i precedes job j and 0 otherwise. These two sets of variables do not depend on the uncertain parameters as they describe the jobs ordering, which is fixed independently from the values taken by the processing times. In contrast, our robust formulations consider that tardiness, t, and job starting time, s, are only defined after realization of uncertainty, so that in a generic form there are infinite number of variables and constraints, each one representing a realization of uncertainty. We represent this in our formulation by introducing an index set W for our uncertainty set U, possibly uncountable, and utilizing a superscript $(^w)$ in the variables and data, meaning there is one for each $w \in W$. We show later in Subsection 4.2.4 that only a finite subset of W is needed to solve the problem.

Finally, notice that we disregard here the time-indexed formulation (e.g. [102]) in which binary variables indicate when jobs are completed. While advanced decomposition algorithms (called SSDP in [120]) have solved efficiently the deterministic counterpart of $1||\sum_j T_j$, the robust counterpart cannot benefit from these techniques because the binary variables would depend on each vector $p \in U$. Specifically, the resulting formulation would have a pseudo-polynomial number of variables *for each* $p \in U$, making SSDP completely unrealistic [56].

4.2.1 Disjunctive constraints formulation

The robust counterpart of the disjunctive constraints formulation is given by the following:

$$
\begin{aligned}
\min \quad & z & & \\
\text{s.t.} \quad & \sum_{j=1}^{n} t_j^w \leq z & \forall\, w \in W & \quad (4.4)\\
& s_j^w + p_j^w - d_j \leq t_j^w & \forall\, j \in N,\, \forall\, w \in W & \quad (4.5)\\
& s_j^w + p_j^w \leq s_i^w + M y_{ij} & \forall\, i, j \in N,\, i < j,\, \forall\, w \in W & \quad (4.6)\\
& s_i^w + p_i^w \leq s_j^w + M(1 - y_{ij}) & \forall\, i, j \in N,\, i < j,\, \forall\, w \in W & \quad (4.7)\\
& s_j^w \geq 0,\, t_j^w \geq 0,\, z \geq 0 & \forall\, j \in N,\, \forall\, w \in W & \\
& y_{ij} \in \{0, 1\} & \forall\, i, j \in N,\, i < j, &
\end{aligned}
$$

where variable z is the overall objective, t_j is tardiness of job j and s_j is start time of job j. Big M is defined as maximum makespan over all $p \in U$ of all jobs. Constraint (4.5) calculates tardiness for job j while constraints (4.6) and (4.7) guarantee right ordering of jobs by the use of disjunctive inequalities. Constraint (4.4) guarantees a worst-case objective value.

In case dominance rules (see Section 4.5) are considered the precedence of jobs information can be used and a new set of constraints is added. Let A_i be the set of jobs known to follow job i. We complement the above formulation with the constraints:

$$
y_{ij} = 1 \qquad \forall\, i \in N,\, j \in A_i.
$$

The information of precedence can be used to eliminate variables in the ordering constraints and mitigate the effect of Big M relaxations present in the formulation.

4.2.2 Sequence-position formulation

The robust counterpart of the sequence-position formulation is given by the following:

$$
\begin{aligned}
\min \quad & z & & \\
\text{s.t.} \quad & \sum_{k=1}^{n} t_k^w \leq z & \forall\, w \in W & \\
& \sum_{i=1}^{n} p_i^w \sum_{u=1}^{k} x_{iu} - \sum_{i=1}^{n} d_i x_{ik} \leq t_k^w & \forall\, k \in N\,, \forall\, w \in W & (4.8)\\
& \sum_{i=1}^{n} x_{ik} = 1 & \forall\, k \in N & (4.9)\\
& \sum_{k=1}^{n} x_{ik} = 1 & \forall\, i \in N & (4.10)\\
& x_{ik} \in \{0,1\},\ t_k^w \geq 0,\ z \geq 0 & \forall\, k, i \in N\,, \forall\, w \in W, &
\end{aligned}
$$

where variable z is the overall objective and t_k is tardiness of job in position k. Constraint (4.8) calculates tardiness for job in position k while constraints (4.9) and (4.10) guarantee that each position is occupied by only one job and each job occupies only one position.

In case dominance rules are considered the precedence of jobs information can be used to reduce the number of sequence-position variables. We define B_i as the set of jobs known to precede job i and then x_{ik}, $i \in N$ and $|B_i| + 1 \leq k \leq n - |A_i|$. In that case constraints (4.8), (4.9) and (4.10) are substituted for:

$$
\begin{aligned}
& \sum_{i=1}^{n} p_i^w \sum_{\substack{u=1 \\ if\ |B_i|<u\leq n-|A_i|}}^{k} x_{iu} - \sum_{\substack{i=1 \\ if\ |B_i|<k\leq n-|A_i|}}^{n} d_i x_{ik} \leq t_k^w \quad \forall\, k \in N\,, \forall\, w \in W \\
& \sum_{k=|B_i|+1}^{n-|A_i|} x_{ik} = 1 \qquad \forall\, i \in N \\
& \sum_{\substack{i \in K \\ K=\{i\,||B_i|<k\leq n-|A_i|\}}} x_{ik} = 1 \qquad \forall\, k \in N
\end{aligned}
$$

We add a new set of constraints to guarantee precedence between jobs:

$$
\sum_{k=|B_j|+1}^{n-|A_j|} k x_{jk} \geq \sum_{k=|B_i|+1}^{n-|A_i|} k x_{ik} + 1 \qquad \forall\, i \in N,\ j \in A_i
$$

4.2.3 Linear ordering formulation

The robust counterpart of the linear ordering formulation is given by the following:

$$
\begin{aligned}
\min \quad & z & & \\
\text{s.t.} \quad & \sum_{j=1}^{n} t_j^w \leq z & & \forall\, w \in W \\
& p_j^w + \sum_{\substack{i=1 \\ i \neq j}}^{n} p_i^w y_{ij} - d_j \leq t_j^w & & \forall\, j \in N,\, \forall\, w \in W \qquad (4.11) \\
& y_{ij} + y_{ji} = 1 & & \forall\, i, j \in N,\, 1 \leq i < j \leq n \qquad (4.12) \\
& y_{ij} + y_{jk} + y_{ki} \leq 2 & & \forall\, i, j, k \in N\, i \neq j, j \neq k, i \neq k \qquad (4.13) \\
& y_{ij} \in \{0,1\},\, t_j^w \geq 0,\, z \geq 0 & & \forall\, i, j \in N,\, i \neq j,\, \forall\, w \in W,
\end{aligned}
$$

where variable z is the overall objective and t_j is tardiness of job j. Constraint (4.11) calculates tardiness for job j while constraints (4.12) and (4.13) guarantee the right ordering of jobs.

In case dominance rules are considered the precedence of jobs information can be used and a new set of constraints is added:

$$y_{ij} = 1 \qquad \forall\, i \in N\,, j \in A_i \qquad (4.14)$$

4.2.4 Row-and-column generation algorithm

To solve our robust integer formulation we use the decomposition algorithm proposed by [128]. We relax the problem into a master problem where each robust constraint is written only for a finite subset $U^0 \subseteq U$. Given a feasible solution to the master problem, we check whether the solution is feasible for each robust constraint by solving adversarial separation problems. In case one or more robust constraints are infeasible, we expand U^0 by one or more vectors and solve the augmented master problem under a row-and-column generation approach.

It turns out that our adversarial separation problem is in fact the problem of, given a sequence of jobs, finding the value of the worst-case realization of uncertainty and verifying if that value is greater than the one provided by the master problem. We can solve that by the methods described in Section 4.4. Since our uncertainty set U is polyhedral, there will be a finite number of extreme solutions to search and the algorithm finitely converges [128].

4.3 Branch-and-bound

The total tardiness problem has been studied through many methods of implicit enumeration, notably of branch-and-bound type. We develop below a branch-and-bound algorithm to solve our robust problem to optimality. The key elements of our branch-and-bound algorithm are: branching, node selection and bound and pruning. We describe each one of them in what follows.

4.3.1 Branching

We create a search tree with no jobs scheduled at the root node. From the root node, n branches lead to n nodes on the first level, each of which corresponds to a particular job being scheduled in the n-th position. Generally, each node at level l in a tree corresponds to a set $J_l \subseteq \{1, \ldots, n\}$ filling the last l positions in a given order. By successively placing each job j ($j \in N \backslash J_l$) in the $|N \backslash J_l|$-th position, $|N \backslash J_l|$ new nodes are created. This is reasonable because all sequences of jobs are feasible in our problem.

4.3.2 Node selection

We make use of a best bound or a depth first approach as search strategy for node selection. For best bound the node selected is the one, among unprocessed nodes, with minimum lower bound. This way we never branch any node whose lower bound is larger than the optimal value. For depth first the node selected is the one, among unprocessed nodes with maximum depth in the search tree. This way we navigate the tree prioritizing the search of new incumbent values.

4.3.3 Bound and pruning

After each branching we prune nodes based on dominance rules described in Section 4.5, on lower bounds when it is greater than the best upper bound calculated at that stage and on optimality conditions when lower bound matches upper bound of a given node.

Upper bound

An initial upper bound, as incumbent solution, is calculated at the root node based on a constructive heuristic algorithm (see Algorithm 2 for its pseudocode). The general idea behind this algorithm is that we shall assign jobs from last position to first. At each position we assign the job that provides the smallest ratio of the least tardiness and the greatest maximum processing time. By doing so, we leave

jobs with greater tardiness or less maximum processing times to be assigned later in the sequencing, where their value of tardiness will decrease or their contribution to partial makespan of that position will be less than other jobs already assigned.

We leverage the calculated dominance rules to create sets of allowable jobs for each position k, $ALJ[k]$. Let SJ be the sequence of jobs already selected, from position n to position $k+1$. Then, we define $ALJ[k] = \{i \in N \backslash SJ \mid A_i \subseteq SJ\}$. This set is never empty, since if all A_i, $i \in N \backslash SJ$, have elements outside SJ, every element in $N \backslash SJ$ is followed by another element in $N \backslash SJ$ which creates a subcycle and that is prevented by construction when calculating the dominance rules (for example see [73] where these cycles are avoided by constructing the transitive closure of the set of known precedence relations immediately after a new relation has been found and by only examining pairs of jobs that are not yet related).

We assign the job i with the least ratio $T_i / \max_{p \in U} p_i$ to each position. This ratio for job i is given by the formula $\frac{\max(0, \max_{p \in U} \sum_{j \in N \backslash SJ} p_j - d_i)}{\bar{p}_i + \hat{p}_i}$. We solve the worst-case evaluation problem (Section 4.4), using the sequence assigned by our heuristic, to calculate the initial upper bound.

An upper bound is also calculated at the last level of the search tree using the sequence of jobs defined for that level, and solving the worst-case evaluation problem.

Lower bound

We detail below the lower bound that can be used to cut part of the branch-and-bound tree. We first recall the deterministic lower bound used by [117], among others, before adapting it to the robust counterpart. The first element used to provide lower bound is based on the idea that the solution for a modified relaxed due date, d_i^{relax}, problem is a lower bound for our original problem.

Observation 1. *Given any* $i \in N$ *and* $\sigma \in X$, $d_i^{relax} \geq d_i \Rightarrow T_i^{relax}(\sigma) = \max(C_i(\sigma) - d_i^{relax}, 0) \leq T_i(\sigma)$.

Using the above property, [117] modifies the due dates in such a way that the optimal schedule becomes easy to provide. Specifically, they apply the following properties:

1. The EDD schedule (ordering jobs by non decreasing due dates) is optimal if $C_{\sigma(k)} \leq d_{\sigma(k)} + p_{\sigma(k)}$ for all positions k [61].

2. The statement above can be relaxed to be true only for positions k where $p_{\sigma(k)} < \max(p_{\sigma(1)}, \ldots, p_{\sigma(k-1)})$, because on the other hand if $p_{\sigma(k)} \geq \max(p_{\sigma(1)}, \ldots, p_{\sigma(k-1)})$ other conditions in [61] assure that jobs $\{\sigma(1), \ldots, \sigma(k-1)\}$ precede job $\sigma(k)$ [117].

We can extend the two statements above to our robust problem, where we verify that if it is valid in a worst-case realization of uncertainty, it will be valid in all cases.

Observation 2. *Given an EDD schedule, if* $\max_{p\in U} C_{\sigma(k-1)} \leq d_{\sigma(k)}$ *for each position* $k > 1$ *when* $\min_{p\in U} p_{\sigma(k)} < \max(\max_{p\in U}(p_{\sigma(1)}), \ldots, \max_{p\in U}(p_{\sigma(k-1)}))$ *then it is optimal.*

We modify the due dates of our original problem so that it satisfies conditions of Observation 2. For the EDD schedule, when $\min_{p\in U} p_{\sigma(k)} < \max(\max_{p\in U}(p_{\sigma(1)}), \ldots, \max_{p\in U}(p_{\sigma(k-1)}))$, for $k > 1$, we make

$$d_{\sigma(k)} \leftarrow \max(\max_{p\in U} C_{\sigma(k-1)}, d_{\sigma(k)})$$

We utilize these due dates to provide a lower bound to our original problem. On the other hand, this relaxed version of our problem has an immediate solution, the EDD sequence.

As already seen, each node level l of our search tree is composed of a set J_l filling the last l positions in a given order. Since the order of the last l positions is defined, we concentrate on an EDD sequencing for the first $n-l$ positions. We order by due date the first $n-l$ positions. We relax minimally their due dates. We concatenate the solution above, using modified relaxed due dates, with the other sequenced jobs of the node, using original due dates and use the worst-case evaluation methods presented in Section 4.4 to find the associated lower bound.

The pseudocode used for our branch-and-bound algorithm is presented in Algorithm 3.

Algorithm 2 Upper bound heuristic

Input ▷ Γ and vectors $\bar{p}, \hat{p}, d$
$SJ \leftarrow$ void ▷ Sequence of selected jobs
for $k = n$ to 1 **do**
  $ALJ[k] = \{i \in N \backslash SJ \mid A_i \subseteq SJ\}$ ▷ Allowable jobs for position k
  $JR_i = \frac{\max(0, \max_{p\in U} \sum_{j\in N\backslash SJ} p_j - d_i)}{\bar{p}_i + \hat{p}_i}$ for $i \in ALJ[k]$ ▷ Store ratio job i
  if $\min_i JR_i = 0$ **then** $SJ \leftarrow SJ \cup \arg\max_{i\in ALJ[k], JR_i=0} \bar{p}_i + \hat{p}_i$
  else $SJ \leftarrow SJ \cup \arg\min_{i\in ALJ[k]} JR_i$ ▷ job selected for position k
end for
Calculate maximum total tardiness for sequence SJ ▷ Worst-case solution
Return solution - sequence of jobs SJ and maximum total tardiness value

Algorithm 3 Branch-and-bound algorithm

Input ▷ Γ and vectors $\bar{p}, \hat{p}, d$
Initialize ▷ Nodes list ← root node, Incumbent solution ← void, Lower bound ← $-\infty$
while There are still nodes to be branched in the Nodes list **do**
  Node Select ▷ Select node based on search criteria
  Prune ▷ by lower bound
  Update Incumbent solution ▷ use best upper bound so far
  Branch node
  Prune ▷ by dominance rules
  Calculate lower and upper bound ▷ of new nodes
  Update Nodes List
end while
Return optimal solution - sequence of jobs and total tardiness value

4.4 Worst-case evaluation

In this section we discuss how to perform an evaluation of the worst-case realization of the uncertainty set given a sequence of jobs. Recall that this evaluation is used to solve the adversarial separation problem in the row-and-column generation algorithm and also used in our branch-and-bound algorithm. This problem, for a given sequence of jobs $\sigma = \{\sigma(k), k = 1, \ldots, n\}$, where k represents a position and $\sigma(k)$ represents the job that occupies that position, is defined by the worst-case total tardiness, T^*_σ, associated with this sequence and given by:

$$T^*_\sigma = \max_{p \in U} \left(\sum_{k=1}^{n} \max \left(0, \sum_{k'=1}^{k} p_{\sigma(k')} - d_{\sigma(k)} \right) \right) \tag{4.15}$$

For this study we adopt, and our algorithm requires, a polyhedral uncertainty set as defined in [31] and introduced in Section 4.1. We assume that maximal deviations of processing times are integers. Also, for the sake of simplicity, we consider in what follows that Γ is a non negative integer. As we will see, this polyhedral uncertainty set allows us to explore some properties that simplify complexity of our algorithms.

Using these assumptions, statement (4.15) reflects a problem with a convex function being maximized over a polytope defined by uncertainty set U. Hence, to define the worst-case robust realization of uncertainty we only have to take into account specific realizations of the uncertainty set given by:

- Extreme points of the polytope. For each job i, we only consider values $\bar{p}_i$ and $\bar{p}_i + \hat{p}_i$
- It is clear that any worst-case realization will use as much budget of uncertainty as possible. Hence we can assume $\sum_i \xi_i = \Gamma$

We work two alternative methods to evaluate a worst-case solution value: one based on dynamic programming and another based on a simple heuristic.

4.4.1 Dynamic programming

We adopt a dynamic programming algorithm developed by [5] for a general class of robust optimization problems. The complexity of this algorithm is $O(n\Gamma\bar{\Phi})$, where $\bar{\Phi}$ is the maximum cumulative processing time deviation allowed, that is $\bar{\Phi} = \max_{p \in U} \sum_{i=1}^{n} \hat{p}_i$. It is favored when processing time deviations are small. We verify that the optimal solution for the adversarial problem only depends on the cumulative deviations of the job processing times. Let us define a value-function $\alpha(k, \gamma, \Phi)$, where $1 \leq k \leq n$, $0 \leq \gamma \leq \Gamma$, $0 \leq \Phi \leq \bar{\Phi}$, as the optimal value of the restricted problem for a set of jobs positions $\{1, \ldots, k\}$ with at most γ deviations and a cumulative deviation Φ. The optimal value of the adversarial problem is defined by $T_\sigma^* = \max_{\Phi \in \{0,\ldots,\bar{\Phi}\}} \alpha(n, \Gamma, \Phi)$. Furthermore, we see that the value-function satisfies the recursion:

$$\alpha(k,\gamma,\Phi) = \begin{cases} \max(0,\ \Phi + \sum_{k'=1}^{k} \bar{p}_{\sigma(k')} - d_{\sigma(k)}) + \alpha(k-1,\gamma,\Phi), & \text{if } \Phi - \hat{p}_{\sigma(k)} < 0 \\ \max(0,\ \Phi + \sum_{k'=1}^{k} \bar{p}_{\sigma(k')} - d_{\sigma(k)}) + \\ \qquad \max(\alpha(k-1,\gamma,\Phi),\ \alpha(k-1,\gamma-1,\Phi-\hat{p}_{\sigma(k)})), & \text{if } \Phi - \hat{p}_{\sigma(k)} \geq 0 \end{cases}$$

Also, the following statements are used to initialize the dynamic programming table:

$$\alpha(1,0,0) = \max(0,\ \bar{p}_{\sigma(1)} - d_{\sigma(1)})$$
$$\alpha(1,\gamma,\hat{p}_{\sigma(1)}) = \max(0,\ \hat{p}_{\sigma(1)} + \bar{p}_{\sigma(1)} - d_{\sigma(1)}), \quad 1 \leq \gamma \leq \Gamma$$
$$\alpha(1,\gamma,\Phi) = -\infty, \text{ for remaining cases}$$
$$\alpha(k,0,0) = \max(0,\ \sum_{k'=1}^{k} \bar{p}_{\sigma(k')} - d_{\sigma(k)}) + \alpha(k-1,0,0), \quad 2 \leq k \leq n$$
$$\alpha(k,0,\Phi) = -\infty, \quad 2 \leq k \leq n,\ 1 \leq \Phi \leq \bar{\Phi}$$

4.4.2 Heuristic

We implement a simple greedy algorithm of complexity $O(\Gamma n^2)$ to obtain a lower bound on T_σ^*. The pseudocode for this heuristic is presented in Algorithm 4. Given a sequence of jobs, we execute an algorithm with Γ iterations. At each iteration we define a job to have its processing time deviated to its maximum. Following are the steps executed:

- At each one of the Γ iterations, we verify total tardiness originated by setting each one of the n jobs processing time to its maximum (if not already

considered in previous iteration as deviated) and pick the one that provided maximum total tardiness.

- If total tardiness is not augmented in relation to the previous iteration we set the processing time of the first job in the sequence not already deviated to its maximum.

Algorithm 4 Worst-case evaluation heuristic

Input ▷ Sequence of jobs: $\sigma = \{\sigma(k), k = 1, \ldots, n\}$, Γ and vectors $\bar{p}, \hat{p}, d$
$D \leftarrow$ void ▷ Set of jobs with deviated processing times
$MAXTT = -\infty$ ▷ Worst-case total tardiness
for $g = 1$ to Γ **do**
$TT_i(D) = \sum_{j=1}^{N} \max(0, (\sum_{k \leq \sigma^{-1}(j)} \bar{p}_{\sigma(k)} + \sum_{\substack{k \leq \sigma^{-1}(j) \\ \sigma(k) \in D \cup \{i\}}} \hat{p}_{\sigma(k)} - d_j))$, for $i \in N \backslash D$
if $\max_i \{TT_i(D)\} > MAXTT$ **then** $D \leftarrow \arg\max \{TT_i(D)\}; MAXTT = \max_i \{TT_i(D)\}$
else $D \leftarrow \arg\min_{i \in N \backslash D} \{\sigma^{-1}(i)\}$
end for
Total tardiness = $MAXTT$
Return Total tardiness

The heuristic can be used as a lower bound when the performance of an exact solution in the algorithm is an issue.

4.5 Dominance rules

Dominance rules have been extensively used in the past in combinatorial optimization problems and specially in scheduling problems [72]. A dominance rule is established in order to reduce the solution space either by adding new constraints to the problem, or by writing a procedure that attempts to reduce the domain of the variables, or by building interesting solutions directly.

One of the main theoretical developments for the total tardiness problem were the dominance rules derived by [61]. The author proved three fundamental theorems that helped establish precedence relations among job pairs that must be satisfied in at least one optimal schedule. These dominance rules are a major component of existing state-of-art algorithms. All the three theorems assume the ordering of jobs by their processing times. Although we can naturally apply these rules to our robust problem for jobs that do not overlap, that is, given two jobs $i, j, \max_{p \in U} p_i < \min_{p \in U} p_j$, there is a tendency to have the applicability of these rules reduced since processing times are uncertain.

Later, [73], working with single machine weighted total tardiness problem, extended these dominance conditions to more general forms where the ordering of processing times are not always necessary, and so, are more adequate to be extended to our robust problem where processing times are uncertain.

Extending the dominance rules defined by [73] to our robust problem is based on the idea that if a rule is valid for a sequence of jobs in a worst-case event realization of uncertainty it will be valid in all realizations of uncertainty. In other words, if a job i precedes job j in a worst-case realization of uncertainty, it will precede in all cases. Hence, this additional restriction can be added to the overall problem.

To extend dominance conditions of [73] to our robust problem we have to consider definitions that follow, where B_i and A_i are sets of jobs known to precede job i and follow i, respectively. Also for notation purposes, for any set Q, $Q \subseteq N$, the notation $P(Q)$ represents $\sum_{i \in Q} p_i$.

- We order two jobs i, j that do not overlap. That is $i < j$ if $\max_{p \in U} p_i < \min_{p \in U} p_j$.
- We define the earliest completion date of a job i, $E_i = \min_{p \in U} (P(B_i) + p_i)$
- We define the latest completion date of a job i, $L_i = \max_{p \in U} P(N - A_i)$

Using the definitions above, conditions of [73] can be restated as follows.

Observation 3. *Job i precedes job j in at least one optimal schedule if at least one of the conditions below are satisfied:*

- $i < j$ *and* $d_i \leq \max(E_j, d_j)$
- $d_j \geq \max(L_i - \min_{p \in U} p_j,\ d_i)$
- $d_j \geq L_i$

By applying these rules successively we can populate, for each job i, the set of jobs known to precede i, B_i and follow i, A_i in some optimal sequence. The idea is that as we run rules above, and pair of jobs are ordered, the sets A_i and B_i will grow, favoring new runs.

These dominance conditions can be incorporated in our MILP formulations as precedence constraints or can be used to prune non optimal node sequences in our branch-and-bound algorithm. They can also be applied dynamically, during branching decisions at each node. If, applied for the subproblem of sequencing jobs $j \in N \backslash J_l$, they identify that there is job $i \in N \backslash J_l$ that precedes a job j, then job i can be eliminated and job j considered for the $|N \backslash J_l|$-th position.

Calculating latest and earliest completion times for each subproblem can, though, add non desired computational complexity. Many proposed algorithms avoid this

additional complexity by applying at each node only a relaxed version of the third condition of Observation 3, called Elmaghraby's lemma [60], where the latest completion time of each job $i \in N \backslash J_l$, L_i, is relaxed to makespan of the subproblem. In other words, if there is a job j that has zero tardiness for the last position ($d_j \geq$ makespan), it can be considered for branching and all other jobs $i \in N \backslash J_l$ eliminated (for examples, see [73] and [100]).

We experiment a compromise between these two previous approaches. We relax the latest and earliest completion dates of all jobs $j \in N \backslash J_l$ by considering maximal ($\max_{p \in U} P(N \backslash J_l)$) and minimal ($\min_{p \in U} P(N \backslash J_l)$) makespan, and setting $L_j^{adj} = \min(L_j, \max_{p \in U} P(N \backslash J_l))$ and $E_j^{adj} = \min(E_j, \min_{p \in U} P(N \setminus J_l))$, where L_j^{adj} and E_j^{adj} are the adjusted latest and earliest completion dates for job j, and L_j and E_j are the latest and earliest completion dates for job j calculated for the original set of jobs N. Observation 3, adjusted, can be used to identify a job $j \in N \backslash J_l$ that succeeds other jobs of set $N \backslash J_l$. If that job j is found, the number of new nodes during branching can be reduced.

Observation 4. *If a job $i \in N \backslash J_l$ satisfies one of the conditions below, for some job $j \in N \backslash J_l$, $i \neq j$, then job i can be eliminated during branching at level l of the search tree.*

- $i < j$ *and* $d_i \leq \max(E_j^{adj}, d_j)$
- $d_j \geq \max(L_i^{adj} - \min_{p \in U} p_j,\ d_i)$
- $d_j \geq L_i^{adj}$

As a remark, any assignment of positions should be in alignment with precedence constraints already generated.

4.6 Implementation and results

4.6.1 Implementation details

Instances We create instances based on the same directives as in [100] to vary hardness of problem solving using parameters R, relative range of due dates and T, average tardiness factor. These parameters are used to define the average and range of variation of due dates. Due date range significantly affects the time performance of algorithms. Due dates widely distributed are easier to solve. The values of R are chosen as $R = \{0.2, 0.4, 0.6, 0.8, 1.0\}$, and the values of T are chosen as $T = \{0.2, 0.4, 0.6, 0.8\}$. With $P = \sum_{i=1}^{n} \bar{p}_i$ and chosen R and T, we generate a non

negative integer due date d_i from the uniform distribution $[P(1-T-R/2), P(1-T+R/2)]$ for each job i.

We define a new parameter G, to control the relative range of variation of uncertainty. This range affects the distance between solution values of one realization of uncertainty to others. A small Γ and large process deviation time potentially produce larger solution distances. The values of G are as $G = \{10, 100\}$. For each job i, an integer processing time $\bar{p}_i$ is generated from the uniform distribution $[1, 100]$ and an integer processing deviation time $\hat{p}_i$ is generated from the uniform distribution $[\bar{p}_i \frac{2}{G}, \bar{p}_i \frac{7}{G}]$. The value of Γ is an integer generated from the uniform distribution $[5 \times 10^{-3} Gn, 9 \times 10^{-3} Gn]$.

Using each combination of R and T and G, we generate 40 instances for problems with 20, 40, 60, 80 and 100 jobs, giving a total of 200 instances tested.

Algorithms Specification Algorithms are coded in Julia [79] using JuMP package and Cplex 12.7. All algorithms run in an Intel CORE i7 CPU 3770 machine. A limit of 9600 seconds of computing time is given for each instance.

All algorithms are tested on the same set of instances. We test MILP formulations (sequence-position, linear ordering and disjunctive constraints) and branch-and-bound algorithms here developed. MILP formulations run under a row-and-column generation method where the separations problems are solved using the dynamic programming algorithm here presented. Each MILP formulation algorithm runs also with the option of using dominance rules to insert precedence constraints.

The branch-and-bound algorithm runs with the options of best bound or depth first search strategy. Upper bounds are calculated at root level and level $n-1$ of our search tree, as described in Section 4.3. Lower bounds are calculated using the dynamic programming method only at level 1. At level $n-1$ the lower bound is equal to the upper bound so that we do not have to recalculate. At other levels of the search tree the lower bound is approximated using the heuristic method of Section 4.4 to improve performance. Dominance rules are used to create precedence between jobs and prune nodes during branching. We also apply dynamically, at each node, the two last conditions of Observation 4.

The name and configuration of each algorithm is presented in Table 4.1.

4.6.2 Comparative performance of the algorithms

Initial tests of our algorithms revealed that the disjunctive MILP formulation algorithm performed much worse, among all instances, when compared to others algorithms, so that we eliminated it from our analyses. We first present in Table 4.2 general results comparing performance of the algorithms for all instances. The $\%Best\ Performance$ measurement indicates that the algorithm $BB1$ presented the

greatest number of instances with best performance, followed by algorithm $LIN1$. Measurement $\%\,NS\,100\,jobs$ indicates that all algorithms were not able to solve the majority of the 100 jobs instances within the time limit. This measurement, together with the presented average solution times, indicate that although $SEQ1$ and $SEQ2$ algorithms had lower $\%Best\ Performance$ measurements, they were able to solve more $\%\,NS\,100\,jobs$ under the time limit and to achieve better $T_{average}$.

We also present Figure 4.1 comparing all different algorithms and using a performance profile [58]. In this figure, the vertical axis points out in percentage, for each algorithm, in how many instances the result was not more than x times - horizontal axis - worse than the best algorithm. For $x = 1$, the indicators replicates the best performance indicators presented in Table 4.2.

Figure 4.1 evidences that each algorithm has different time variation characteristics. In particular, algorithm $LIN1$ and $SEQ1$ present the highest ascending slopes, indicating that, although they are not best performers as measured by $\%Best\ Performance$, they present the characteristic of less solution time variance among all algorithms.

We then present Table 4.3 where the same indicators are listed, but now grouped by special instances. We follow concepts defined in [100] and group instances based on their R, relative range of due dates and T, average tardiness factor. Instances with ratio T/R greater than 1 are classified as "High hardness" and classified as "Low hardness" otherwise. In general "Low hardness" instances will be more completely classified by dominance rules as stated in [100]. We also group instances by their G value. Instances with $G = 10$ are classified as "Large uncertainty range" and as "Small uncertainty range" otherwise. "Large uncertainty range" instances are the ones that, in general, have large solution value distance between one realization of uncertainty to others.

In our analysis below we analyze best performance by privileging the algorithms that are able to solve a greater percentage of instances in less time when compared to other algorithms.

Analysis of Table 4.3 indicates that algorithm $SEQ2$ has best performance for "High hardness" instances while algorithm $BB1$ has best performance for "Low hardness" instances and "Large uncertainty range" instances.

These results so far are expected since, on one side, the branch-and-bound algorithm relies heavily on dominance conditions to prune nodes and this is favored in "Low hardness" instances. On the other hand, our MILP RCG algorithms are not favored by "Large uncertainty range" instances, since in general it will require more calls to separation problems.

This effect can be better verified in Figure 4.2 where we present performance profiles for each group of instances. In general, now grouped by special instances,

the best performers algorithms show consistency of performance along the x-axis. It evidences, for "High hardness" instances, that when characteristics as dominance conditions are not favored, the sequence-position formulation is best performer. Algorithm $LIN1$ show best performance for "Small uncertainty range" instances.

These results suggest that, in general, if an instance is not well defined by dominance rules, a "High hardness" instance, $SEQ1$ is favored. If not, if an instance is well defined by dominance rules, a "Low hardness" instance, the characteristic of uncertainty range will define the algorithm to be favored: "Large uncertainty range" instances are favored by $BB1$ and, "Small uncertainty range" instances are favored by $LIN1$.

In Figure 4.3 we present different performance profiles, now comparing each MILP RCG method among their different configurations. We can observe that both sequence-position and linear ordering formulations had better performance when precedence constraints, based on dominance rules, were added. The effect of precedence constraints was most pronounced in $LIN1$ algorithm. This can be partially explained because the precedence constraints added for the $LIN2$ algorithm (see equation 4.14) are effective to restrict the search of the CPLEX linear programming algorithm to a reduced set of extreme points.

Table 4.4 presents average time performance of each MILP RCG method and configuration. Time performance is split between master and adversarial problems. We also present average number of iterations. These results evidence that the critical step for these algorithms is the master problem resolution, even with the addition of precedence constraints.

Figure 4.4 presents performance profile for our 2 branch-and-bound algorithms. Algorithm $BB1$, based on a depth first search strategy, clearly outperforms algorithm $BB2$, based on a best bound search strategy. To analyze this effect we present in Table 4.5 detailed results for our branch-and-bound algorithms. It shows that the dominance rules were effective to prune nodes in both strategies. It also shows that the best bound strategy was not successful as the depth first strategy to prune nodes by its lower bound. In fact, by the way our branch-and-bound algorithms were implemented, lower bound values improves as the algorithm reaches the leaves of the tree and that favors the depth first strategy. It is also a consequence of our choice of a lower bound algorithm that is easy to calculate but not tight. On the other hand, depth first search strategy is also able to find an upper bound more quickly, which helps to improve performance.

Algorithm Name	Method	Dominance Rules	Best Bound Strategy	Depth First Strategy
Seq1	Sequence MILP RCG	Yes	-	-
Seq2	Sequence MILP RCG	No	-	-
Lin1	Linear MILP RCG	Yes	-	-
Lin2	Linear MILP RCG	No	-	-
BB1	Branch-and-bound	Yes	No	Yes
BB2	Branch-and-bound	Yes	Yes	No

RCG meaning row-and-column generation

Table 4.1: Algorithms

4.6.3 Assessing the robustness

We assess below the costs provided by different models under different uncertainty sets on two instances with 20 jobs. The six models considered are related to the robust model used and to the value of $\Gamma \in \{0, 5, 10, 20\}$. The model with $\Gamma = 0$ is the deterministic model that ignores uncertainty, while the one with $\Gamma = 20$ is the deterministic model that is completely risk-averse and overestimates all parameters. The other values of Γ model intermediate risk aversions by using either the robust model studied here, or the simpler static model obtained by adding the constraints $T_i(p, \sigma) = T_i(p', \sigma)$ for each $i \in N$ and $p, p' \in U$. Let σ^Γ and σ^Γ_{stat} be the solutions obtained by the robust models for the value $\Gamma \in \{10, 15\}$, and denote similarly the deterministic solutions by σ^0_{det} and σ^{20}_{det}. We report on Figure 4.5 the costs

$$T^\Gamma(\sigma) = \max_{p \in U} \sum_{i \in N} T_i(p, \sigma)$$

on two instances that illustrate the general patterns that can be observed.

A "Large uncertainty range" instance, where there are large solution value distances between one realization of uncertainty to others is presented. For this instance, varying the level of conservativeness, that is, varying Γ and the number of jobs processing times that can vary, have significant impact on total tardiness. From Figure 4.5 we see that the nominal solution, as well as static solutions, are suboptimal for $\Gamma \in \{1, \ldots, 7\}$. Specifically, for these values of Γ, σ^5 is roughly 20% cheaper than σ^5_{stat}, and the ratio increased when considering the other models. For larger values of Γ, the cheapest solution is σ^{20}_{det}, with the robust solutions σ^{10} and σ^{10}_{stat} being roughly 5% more expensive. We also see that the deterministic solution σ^0_{det} behaves extremely badly as soon as $\Gamma > 1$. To summarize, analyzing these two figures show that the robust solution σ^5 should be preferred because it is never far from being the cheapest one.

A "Small uncertainty range" instance example is also presented. For this instance, varying level of conservativeness does not have a great impact on total tardiness.

Indicator	Seq1	Seq2	Lin1	Lin2	BB1	BB2
% Best Performance	37	40	41	29	60	26
$\%\, NS\, 100\, jobs$	55	61	72	72	77	77
$T_{average}$	3351.92	3833.10	4208.73	5648.02	5189.01	6404.46

% Best Performance is percentage of total instances where algorithm was best in time performance
$\%\, NS\, 100\, jobs$ is percentage of total 100 jobs instances not solved within time limit
$T_{average}$ is average solution time (s)

Table 4.2: Performance of algorithms for all instances

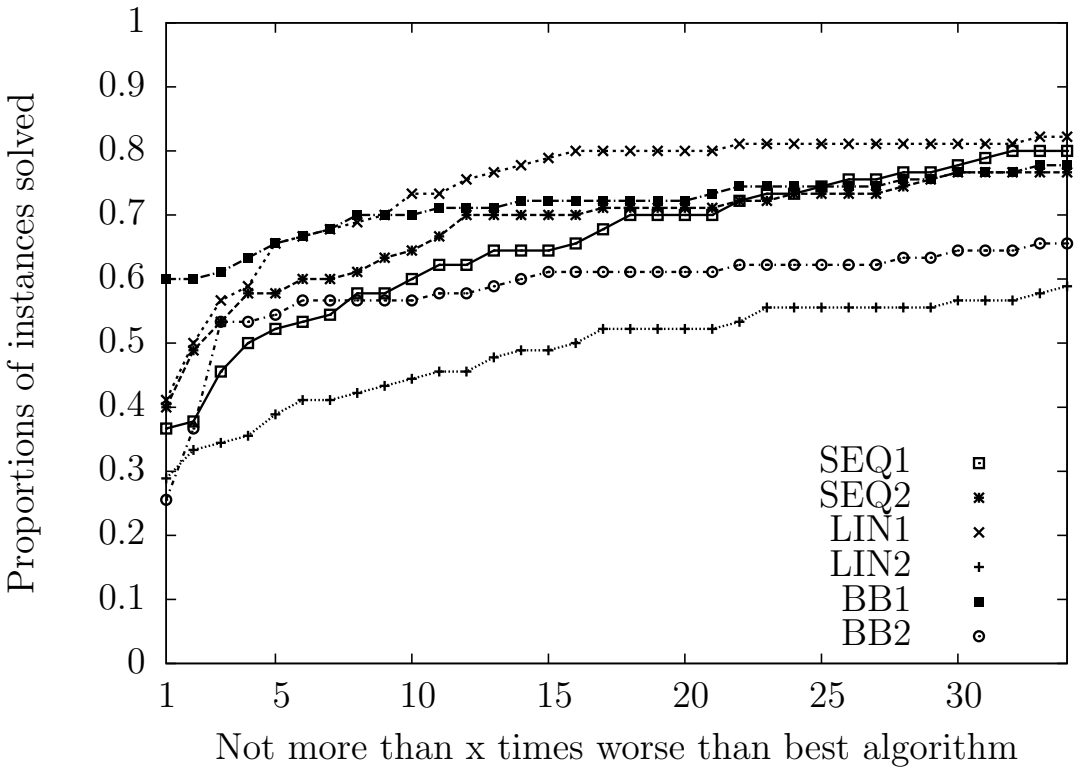

Figure 4.1: Performance profile among all instances

Instances group	Indicator	Seq1	Seq2	Lin1	Lin2	BB1	BB2
High Hardness	% Best Performance	50	58	48	33	38	30
	T_{median}	537.25	575.62	9600	9600	9600	9600
	$\%\, NS\, 100\, jobs$	62	62	100	100	100	100
Low Hardness	% Best Performance	26	26	36	26	78	22
	T_{median}	50.27	64.72	7.55	39.48	2.29	22.79
	$\%\, NS\, 100\, jobs$	50	60	50	50	50	60
Large Uncertainty Range	% Best Performance	58	40	47	40	78	40
	T_{median}	237.34	1808.08	9600	9600	569.78	9600
	$\%\, NS\, 100\, jobs$	67	67	78	78	78	78
Small Uncertainty Range	% Best Performance	16	40	36	18	44	11
	T_{median}	244.76	160.72	77.65	426.41	192.00	585.52
	$\%\, NS\, 100\, jobs$	44	55	66	66	66	77

% Best Performance is percentage of total instances of a group where algorithm was best in time performance
T_{median} is median solution time (s) considering all instances of a group
$\%\, NS\, 100\, jobs$ is percentage of total 100 jobs instances of a group not solved within time limit

Table 4.3: Performance of algorithms, grouping by special instances

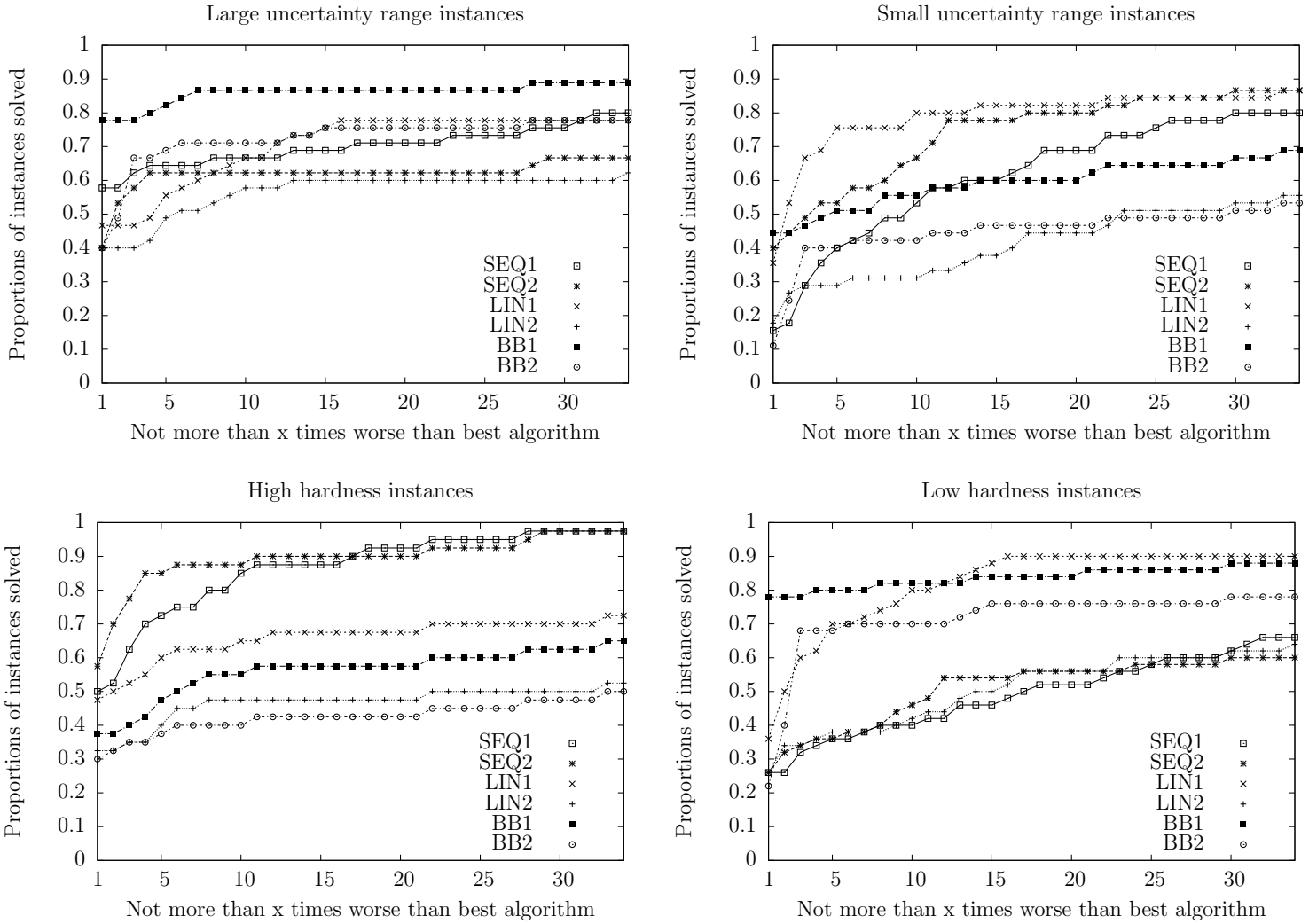

Figure 4.2: Performance profile of algorithms, grouping by special instances

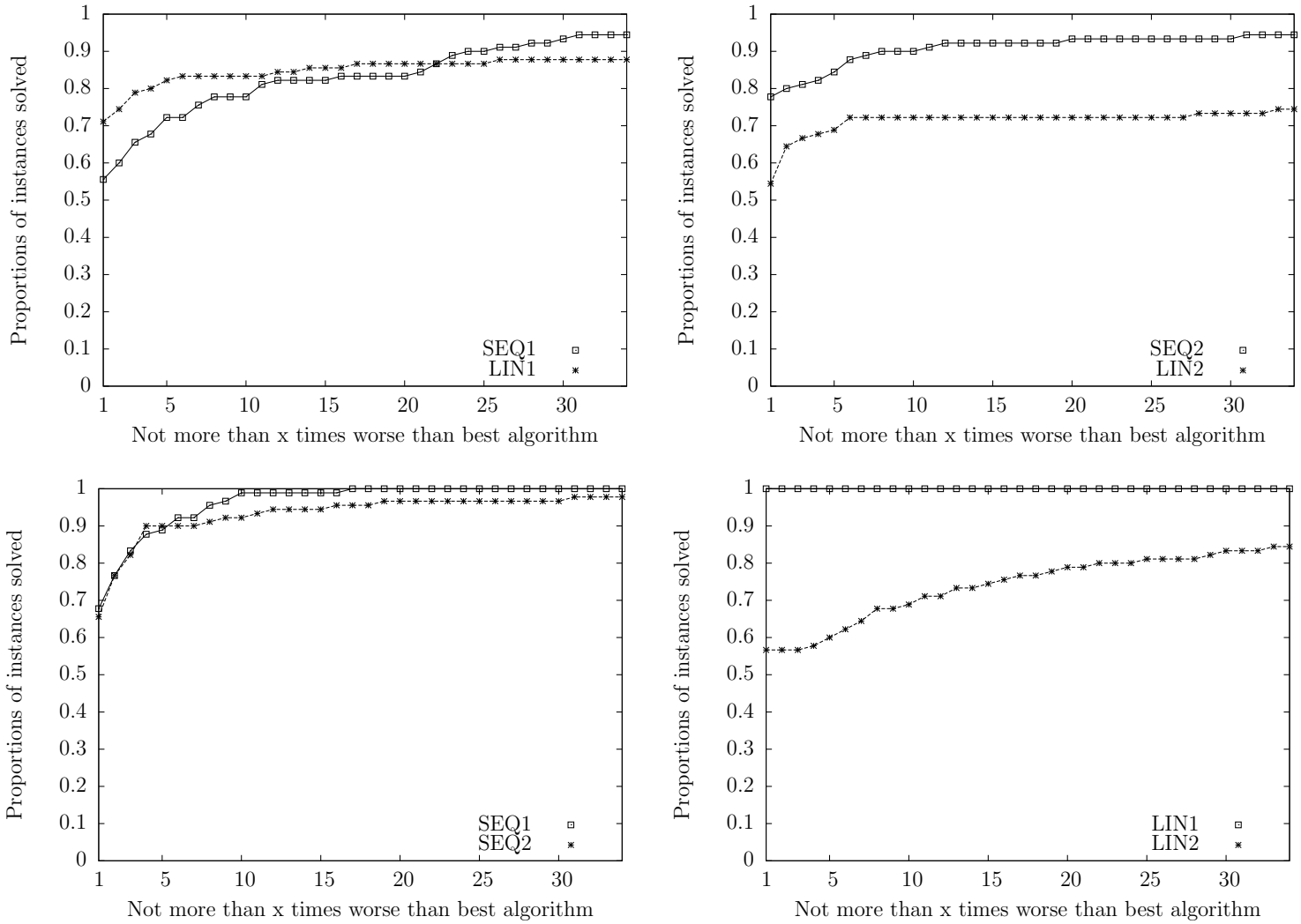

Figure 4.3: Performance profile MILP methods and configurations

Formulation	Dominance(Yes/No)	Master Problem Time*	Adversarial Problem Time*	Number of Iterations*
Linear ordering	No	5637.55	10.47	3
Linear ordering	Yes	4195.68	13.05	3
Sequence-position	No	3802.48	30.62	4
Sequence-position	Yes	3287.21	64.71	4

** All measures presented are average values among all instances*

Table 4.4: MILP algorithms - all instances by method and configuration

Search strategy	Nodes visited*	% Nodes pruned by dominance*	% Nodes pruned by lower bound*	Solution time*	% Solution gap* **
Depth first	278019	72	57	5189.01	15
Best bound	452322	97	0	6404.46	46

** All measures presented are median values among all instances*

*** Calculated for non optimal solutions as percentage difference to best solution among all algorithms*

Table 4.5: Branch-and-bound algorithms - all instances by search strategy

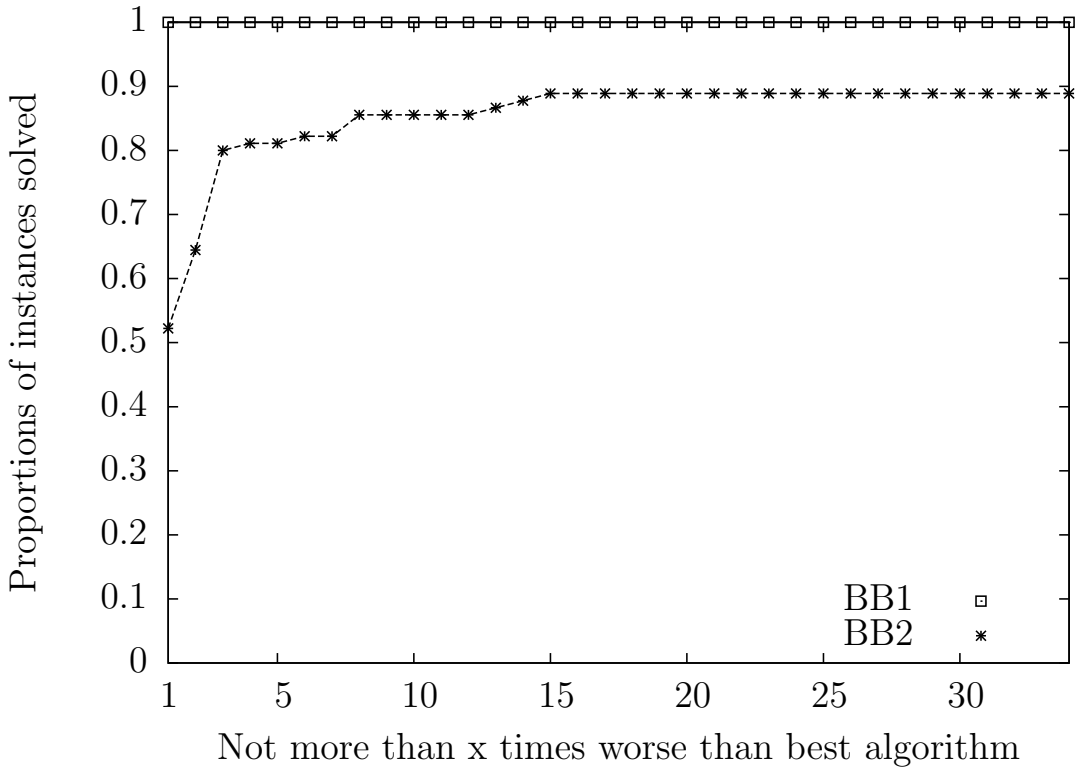

Figure 4.4: Algorithms BB performance profile among all instances

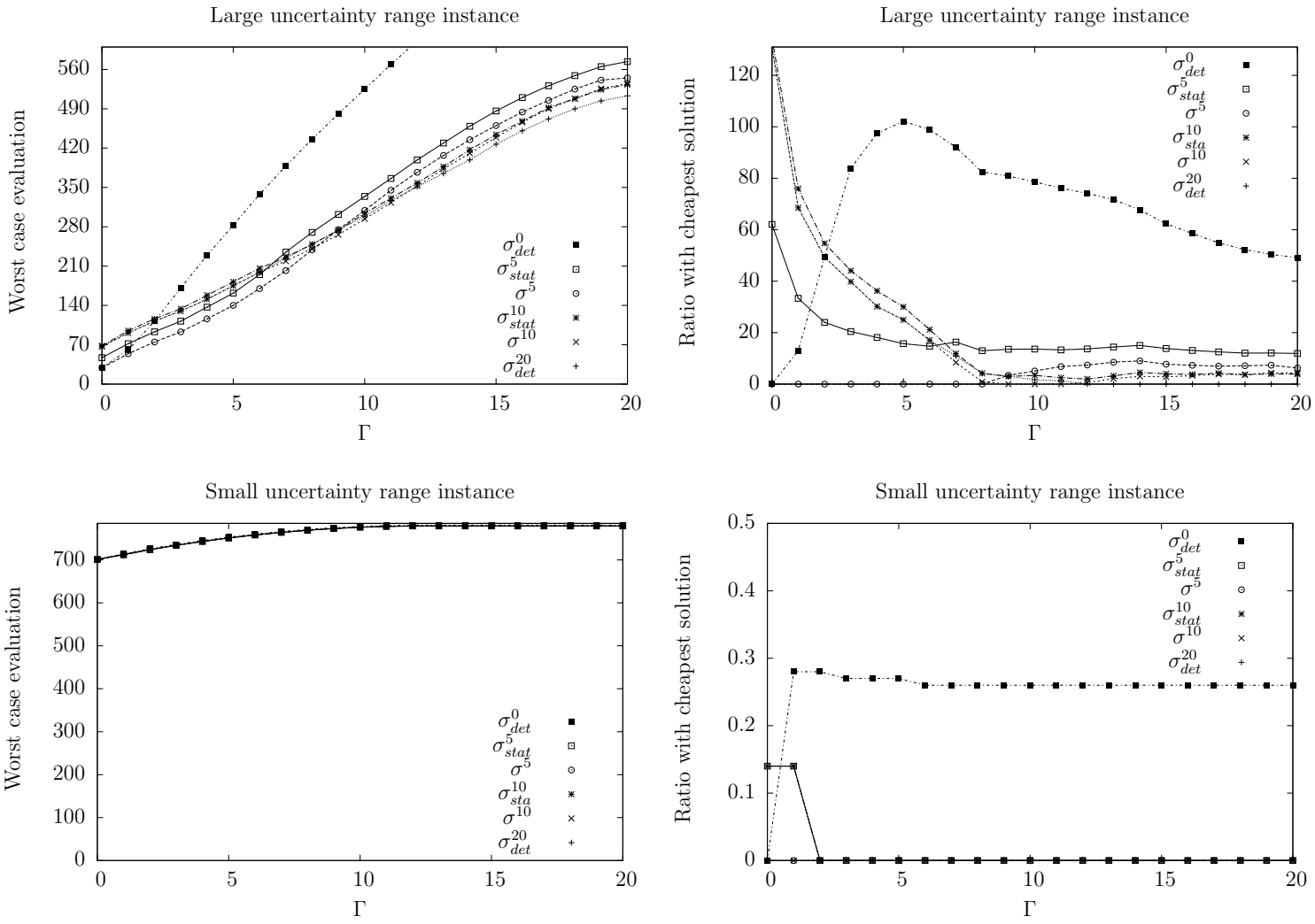

Figure 4.5: Worst-case evaluation for robust and nominal solutions

Chapter 5

Min-max-min robustness for combinatorial problems with polyhedral uncertainty

5.1 Introduction

In this study we consider two-stage robust combinatorial problems, where the decision maker can react to a scenario by choosing from a finite set of first-stage decisions. Besides being an alternative to reduce conservatism when compared to decisions made before the scenario is known, it is a viable alternative to otherwise intractable two-stage problems.

Alternatives to reduce conservatism of the robust optimization approach have been focus of research since the introduction of its concept in [17] and [18]. The challenge resides in maintaining the potential tractability of problems formulated with this classical robust formulation approach.

One alternative to reduce conservatism, named adjustable robust optimization, was introduced in [19] where some of the variables, named wait-and-see variables, are defined only after part of the uncertainty is revealed. Unfortunately, having wait-and-see variables depending dynamically on uncertainty can turn the problem intractable. A natural solution is to restrict the wait-and-see variables to special functions of uncertainty, defined as decision rules. This approximation can turn the problem tractable and different decision rules have been developed.

In [26] the authors develop the finite adaptability decision rules. Here the wait-and-see variables are piecewise constant functions of uncertainty. Thinking in terms of two-stage decision problems, the idea of finite adaptability is to calculate a fixed number k of second-stage solutions, and then committing to one of them only after seeing the realization of the uncertainty. At least one of the k solutions must be

feasible regardless of the realization of the uncertainty.

In [69] the authors extend finite adaptability for two-stage robust binary problems. They experiment the case for uncertainty only in objective parameters and show that their min-max-min equivalent formulation can be solved exactly if one considers $k = n + 1$, where n is the dimension of the second-stage vector variable involved. They propose a reformulation as a mixed-integer linear program for non empty bounded polyhedron uncertainty sets. More recently, in [116], a branch-and-bound algorithm is proposed that solves a sequence of scenario-based k-adaptability problems over monotonically increasing k scenario subsets. At each iteration, a separation problem identifies a new scenario to be added to these sets.

In [38] they consider finite adaptability for the case where no first-stage variables exist. They apply it for solving combinatorial optimizations problems with uncertain objective functions. A fixed number k of feasible solutions is computed such that the respective best of them is optimal in the worst-case. A set of candidate solutions are calculated in a potentially expensive preprocessing phase and then the best solution out of this set is selected in real-time, once the actual scenario is known. The idea behind this approach is that this set of candidate solutions can be calculated only once and the choice of best solution can be done in a posterior dynamic phase that will depend on the scenarios considered. Their k-adaptability approach leads to a min-max-min of the form

$$\min_{x^{(1)},\ldots,x^{(k)} \in X \subset \{0,1\}^n} \max_{c \in U} \min_{i \in \{1,\ldots,k\}} c^T x^{(i)} \qquad (M^3)$$

They assume that X is given implicitly by a linear optimization oracle. The authors prove that (M^3) can be solved in polynomial time for $k \geq n + 1$ and for a convex uncertainty set U if the equivalent deterministic problem can be solved in polynomial time.

In [41] the authors further extend the work in [38] and consider a min-max-min robust combinatorial problem for the case that the set of possible scenarios is described through a budgeted uncertainty set, distinguishing between discrete and convex variants. For the former case, hardness results and a pseudopolynomial algorithm are presented. For the latter case, they identify cases that can be solved in polynomial time and derive heuristic and exact solution methods.

In this study we propose a new formulation and different algorithmic implementations to solve problem (M^3) for a polyhedral uncertainty set U.

In particular, the contributions of this study are:

- We introduce a new extended formulation for problem (M^3) with tight linear programming relaxation.
- We develop different row-and-column generation algorithmic implementations

to solve the introduced extended formulation. In particular, we experiment variations where new variables and constraints are generated on the fly during a branch-and-bound node resolution. We also experiment an initial heuristic and relaxations bounds.

- We perform computational experiments using the 0-1 knapsack problem where the algorithmic implementations are compared using instances randomly generated.

5.2 Formulations

In the next two subsections we present different formulations to solve problem (M^3). From now on we consider that the cost vector c varies within a polyhedral uncertainty set given by $U = \{c \in \mathbb{R}^n \mid Ac \leq b\}$, where $A \in \mathbb{R}^{w \times n}$ and $b \in \mathbb{R}^n$. We also consider that X, the set of feasible solutions, is given implicitly by a linear optimization oracle. For the formulations presented next, consider first an arbitrary finite subset $U' \subset U : U' = \{c^1, \ldots, c^m\}$.

5.2.1 Compact formulation

We first present an assignment-based formulation. Let binary variables p_{ij} define an assignment between the scenarios and the solutions where $p_{ij} = 1$ if and only if $x^{(i)}$ has the minimal objective value over all $x^{(1)}, \ldots, x^{(k)}$ in scenario c^j.

We can reformulate problem (M^3) as

$$
\begin{aligned}
\max \quad & \omega & & \\
\text{s.t.} \quad & \omega \leq \sum_{i=1}^{k} p_{ij} (c^j)^\top x^{(i)} & & \forall\, j \in \{1, \ldots, m\} \\
& \sum_{i=1}^{k} p_{ij} = 1 & & \forall\, j \in \{1, \ldots, m\} \\
& p_{ij} \in \{0, 1\} & \forall\, i \in \{1, \ldots, k\}, & \forall\, j \in \{1, \ldots, m\} \\
& x^{(i)} \in X & & \forall\, i \in \{1, \ldots, k\},
\end{aligned}
\tag{Master}
$$

where the product $p_{ij} x^{(i)}$ can be linearized introducing the additional variables

$px_{ij} = p_{ij}x^{(i)}$ and adding the classical constraints:

$$
\begin{aligned}
\max \quad & \omega \\
\text{s.t.} \quad & \omega \leq \sum_{i=1}^{k}\sum_{\ell=1}^{n} px_{ij\ell}c_\ell^j && \forall\, j \in \{1,\dots,m\} \\
& \sum_{i=1}^{k} \varphi_{ij} = 1 && \forall\, j \in \{1,\dots,m\} \\
& px_{ij\ell} \leq x_\ell^{(i)} && \forall\, i \in \{1,\dots,k\}, \forall\, j \in \{1,\dots,m\}, \forall\, \ell \in \{1,\dots,n\} \\
& px_{ij\ell} \leq p_{ij} && \forall\, i \in \{1,\dots,k\}, \forall\, j \in \{1,\dots,m\}, \forall\, \ell \in \{1,\dots,n\} \\
& px_{ij\ell} \geq x_\ell^{(i)} + p_{ij} - 1 && \forall\, i \in \{1,\dots,k\}, \forall\, j \in \{1,\dots,m\}, \forall\, \ell \in \{1,\dots,n\} \\
& p_{ij} \in \{0,1\} && \forall\, i \in \{1,\dots,k\}, \forall\, j \in \{1,\dots,m\} \\
& x^{(i)} \in X && \forall\, i \in \{1,\dots,k\}.
\end{aligned}
$$

(LinearMaster)

The main bottleneck of formulation (LinearMaster) lies in its weak linear programming formulation relaxation. Another weakness of (LinearMaster) lies in the symmetry among optimization vectors $x^{(i)}$. Symmetry arises when variables can be permuted without affecting the structure of the problem. For integer programming using branch-and-bound techniques as resolution method, the set of feasible solutions is partitioned as part of the technique, forming more easily-solved subproblems. The presence of symmetry means that many of these subproblems are equivalent. Only one member of each collection of equivalent subproblems needs to be solved. Failure to recognize that many subproblems are symmetric results in a waste of computational effort that can render an instance unsolvable by branch-and-bound (see [92] for details). One way to tackle with the symmetry issue is to exploit its knowledge and introduce symmetry breaking linear constraints. These constraints can, for instance, restrict the feasible region in order to avoid possible permutations. Another way is by reformulating (M^3) as will be detailed in next section.

5.2.2 Extended formulation

To improve the weak linear programming relaxation of (LinearMaster), we present below an extended formulation having possibly exponential numbers of variables and constraints. Then, we show how to handle these implicitly using a variant of Dantzig-Wolfe reformulation for extended formulations, see for instance [104].

Let us rewrite X as an enumerated set $X = \{x_s, \forall\, s \in \{1,\dots,r\}\}$ and let $\kappa_s^j = (c^j)^\top x_s$. Let binary variable z_{sj} be equal to 1 iff we affect solution s to scenario j, and y_s be equal to 1 iff solution s is used at all. The extended formulation is

provided below.

$$
\begin{aligned}
\max \quad & \omega \\
s.t. \quad & \omega \leq \sum_{s=1}^{r} \kappa_s^j z_{sj} \quad \forall\, j \in \{1,\dots,m\} \\
& \sum_{s=1}^{r} z_{sj} = 1 \quad \forall\, j \in \{1,\dots,m\} \\
& \sum_{s=1}^{r} y_s = k \\
& z_{sj} \leq y_s \quad \forall\, j \in \{1,\dots,m\}, \forall\, s \in \{1,\dots,r\} \\
& y, z \text{ binary.}
\end{aligned}
\tag{StrongMaster}
$$

We explain below how to handle the large number of constraints and variables of (StrongMaster) using a Dantzig-Wolfe reformulation. Let us rewrite the integral polytope $\Upsilon_s = \{(y_s, z_s) \in [0,1]^{m+1} : z_{sj} \leq y_s, \forall\, j \in \{1,\dots,m\}\}$ as the enumerated set $\{(y_s^q, z_s^q), \forall\, q \in \{1,\dots,t\})\}$ for each $s \in \{1,\dots,r\}$. We reformulate (StrongMaster) by replacing the last group of constraints of (StrongMaster) with convex combinations of the elements of Υ_s, obtaining

$$
\begin{aligned}
\max \quad & \omega \\
s.t. \quad & \omega \leq \sum_{s=1}^{r} \sum_{q=1}^{t} \kappa_s^j z_{sj}^q \lambda_s^q \quad \forall\, j \in \{1,\dots,m\} \quad (\alpha) \\
& \sum_{s=1}^{r} \sum_{q=1}^{t} z_{sj}^q \lambda_s^q = 1 \quad \forall\, j \in \{1,\dots,m\} \quad (\beta) \\
& \sum_{s=1}^{r} \sum_{q=1}^{t} y_s^q \lambda_s^q = k \quad (\delta) \\
& \lambda \text{ binary,}
\end{aligned}
\tag{RMP}
$$

where the dual variables are indicated into parentheses and the multiplicity constraints $\sum_{q=1}^{t} \lambda_s^q = 1, \forall\, s \in \{1,\dots,r\}$ have been omitted, see the next result.

Lemma 1. *Let $\overline{\lambda}$ be a feasible solution to* (RMP)*. For each $s \in \{1,\dots,r\}$, either $\sum_{q=1}^{t} \overline{\lambda}_s^q = 1$ or we can define $\tilde{\lambda}$ such that $\sum_{q=1}^{t} \tilde{\lambda}_s^q = 1$, $\sum_{q=1}^{t} y_s^q \overline{\lambda}_s^q = \sum_{q=1}^{t} y_s^q \tilde{\lambda}_s^q$ and $\sum_{q=1}^{t} z_{sj}^q \overline{\lambda}_s^q = \sum_{q=1}^{t} z_{sj}^q \tilde{\lambda}_s^q$ for each $j \in \{1,\dots,m\}$.*

Proof. Consider $s \in \{1,\dots,r\}$ and let us define the vectors $(0,\dots,0)$ and $(1,0,\dots,0)$ as (y_s^1, z_s^1) and (y_s^2, z_s^2), respectively. The constraints of (RMP) related to α imply that

$$\sum_{q=3}^{t} \lambda_s^q \leq 1.$$

Suppose $\lambda_s^q = 1$ and $\lambda_s^{q'} = 1$. If $y_s^q = 0$ (resp. $y_s^{q'} = 0$) then we can set $\lambda_s^q = 0$ (resp. $\lambda_s^{q'} = 0$) without affecting the solution. Hence, we consider next that $y_s^q = y_s^{q'} = 1$. We take q^* such that $y_s^{q^*} = 1$ and $z_{sj}^{q^*} = \max(z_{sj}^q, z_{sj}^{q'})$. Hence, we can construct a new solution λ such that $\lambda_s^q = \lambda_s^{q'} = 0$ and $\lambda_s^q = 1$. If $\sum_{q=3}^{t} \lambda_s^q < 1$, then we can define $\tilde{\lambda}_s^q = \lambda_s^q$ for each $q \geq 3$, $\tilde{\lambda}_s^0 = 0$ and $\tilde{\lambda}_s^0 = 1$, proving the result. □

Let $\overline{\alpha}$, $\overline{\beta}$, and $\overline{\delta}$ be the optimal dual variables of the linear programming relaxation of (RMP). Following the classical approach (e.g. [124]), we obtain the pricing problem

$$\min_{q\in\{1,\ldots,t\},s\in\{1,\ldots,r\}} \sum_{j=1}^{m} (\overline{\beta}_j + \overline{\alpha}_j \kappa_s^j) z_{sj}^q + \overline{\delta} y_s^q = \min_{q\in\{1,\ldots,t\},s\in\{1,\ldots,r\}} \sum_{j=1}^{m} (\overline{\beta}_j + \overline{\alpha}_j (c^j)^\top x_s) z_{sj}^q + \overline{\delta} y_s^q. \tag{5.1}$$

The minimum over s (resp. q) can be rewritten by turning x (resp. y_j and z) to binary optimization variables, obtaining the following quadratic integer program

$$\begin{aligned} \min \quad & \sum_{j=1}^{m} (\overline{\beta}_j + \overline{\alpha}_j (c^j)^\top x) z_j + \overline{\delta} y \\ s.t. \quad & z_j \leq y \quad \forall\, j \in \{1, \ldots, m\} \\ & x \in X \\ & y, z \text{ binary.} \end{aligned}$$

The main advantage of reformulation (RMP) is that solution s only appears in the formulation whenever an index $q > 1$ is generated. Therefore, the approach can generate on the fly the large number of variables indexed by s. Finally, it is well-known that an alternative to generating variables λ in (RMP) amounts to generate variables (y, z) directly in formulation (StrongMaster), together with the subproblem constraints $z_{sj} \leq y_s, \forall\, j \in \{1, \ldots, m\}$, see [104] for a formal description of that alternative. Both approaches would provide the same lower bound because the Dantzig-Wolfe decomposition is applied to the integral polytopes Υ_s.

5.3 Row-and-column generation algorithm

The formulations developed in the previous Section 5.2 present exponentially many variables and constraints when considering all scenarios and solutions. Typically, only a fraction of these variables and constraints are needed to prove optimality. Hence, we use a row-and-column generation algorithm to solve them. Row-and-column generation is an indispensable tool in integer optimization to solve a mathematical program by iteratively adding the variables and constraints of the model. Even though the method is simple in theory there are many algorithmic choices that

can be done. Here we discuss the approach given to the formulations developed and in Section 5.5 we elaborate on the different algorithmic implementations that were tested.

The first ingredient of our approach, described in Algorithm 5 is to solve (M^3) for the starting set $U' \subset U$ and to iteratively add a new scenario, if it exists, that invalidates the solution. This can be done by calculating the scenario that provides the worst case objective value for the optimal solution calculated with U' and verifying if it belongs to U'.

Algorithm 5 First approach to row-and-column generation algorithm

1: Choose any $c^* \in U$ and set $U' = \emptyset$
2: **repeat**
3: add c^* to U'
4: calculate optimal solution of (M^3) with U'
5: calculate worst case c^* for optimal solution
6: **until** $c^* \in U'$

It is easy to see that Algorithm 5 calculates an optimal solution of problem (M^3). Since the the uncertainty set that comprises the feasible region of the subproblem solved in step 5 is polyhedral, we only have a finite number of basic feasible solutions. We can then only generate a finite number of scenarios and therefore the algorithm terminates in a finite number of steps.

This generic approach can be further customized depending on the formulation used to solve it, as we describe in the next subsections.

5.3.1 Compact Formulation

Using formulation (LinearMaster), we can iteratively add a new scenario which is the optimal solution of subproblem

$$\min \left\{ z \ : \ z \geq c^\top x^{(i)} \quad \forall\, i \in \{1, \ldots, m\},\ c \in U \right\}, \tag{Slave}$$

where the value of $x^{(1)}, \ldots, x^{(k)}$ is the optimal solution calculated in step 4.

In theory, Algorithm 5 could typically be improved by embedding the generation of the inequalities and variables inside a single branch-cut-and-price algorithm solving (LinearMaster). This is a classical speed-up technique in Benders decomposition where one only generates constraints along the iterations (e.g. [64]). While this approach works well for Benders' decomposition, its efficiency is far more limited when variables also need to be generated, because the most efficient commercial solvers cannot handle the dynamic generation of variables.

5.3.2 Extended Formulation

Using formulation (StrongMaster), we can iteratively add new scenarios as we have done using the compact formulation in the previous subsection. Unlike the compact formulation, the extended formulation will benefit from using branch-and-cut-and-price algorithms. Hence, we will also need to generate new scenarios whenever y_s^*, the optimal solution calculated in step 4, is fractional. In this case the scenario separation subproblem, since variables y_s^* are relaxed and where x_s^* are the associated enumerated solutions, is given by:

$$\begin{aligned} \min_{c \in U} \max_{z} \quad & \sum_{s=1}^{r'} \sum_{l=1}^{N} c_l x_{sl}^* z_s \\ s.t. \quad & \sum_{s=1}^{r'} z_s = 1 \\ & z_s \leq y_s^* \qquad \forall\, s \in S' \end{aligned} \tag{5.2}$$

where S' represents the subset of enumerated solutions being considered.

The inner maximization problem can be dualized and the equivalent scenario relaxed separation subproblem is given by:

$$\begin{aligned} \min_{c \in U, u, v} \quad & u + \sum_{s=1}^{r'} y_s^* v_s \\ s.t. \quad & u + v_s \geq \sum_{l=1}^{N} c_l x_{sl}^* \quad \forall\, s \in S' \\ & v_s \geq 0 \qquad\qquad\qquad \forall\, s \in S' \end{aligned} \tag{5.3}$$

5.4 Local search heuristic

We develop a local search heuristic that can be used to calculate an initial incumbent solution. This local heuristic is based on the work developed in [41], which we extend to a general polyhedral uncertainty set. They introduce a variable z to express the inner minimization problem of (M^3). This leads to a min-max problem defined as

$$\begin{aligned} & \min_{x^{(1)},\ldots,x^{(k)} \in X \subset \{0,1\}^n} \max_{c \in U} \max_{z} \left\{ z \mid z \leq c^T x^{(i)}\, \forall\, i \in \{1,\ldots,k\} \right\} = \\ & \min_{x^{(1)},\ldots,x^{(k)} \in X \subset \{0,1\}^n} \max_{c,z} \left\{ z \mid c \in U, z \leq c^T x^{(i)}\, \forall\, i \in \{1,\ldots,k\} \right\}. \end{aligned} \tag{5.4}$$

The inner maximization problem of (5.4) can be dualized and we arrive to a nonlinear formulation, where A_{it} and b_i as elements of the matrix A and the vector b, respectively, that defines the uncertainty set U. It is given by

$$
\begin{aligned}
\min \quad & \sum_{i\in\{1,\dots,w\}} b_i\gamma_i \\
s.t. \quad & \sum_{i\in\{1,\dots,w\}} A_{it}\gamma_i \geq \sum_{j\in\{1,\dots,k\}} x_t^{(j)}\alpha_j \quad \forall\, t\in\{1,\dots,n\} \\
& \sum_{j\in\{1,\dots,k\}} \alpha_j = 1 \\
& x^{(j)} \in X \qquad\qquad\qquad\qquad \forall\, j\in\{1,\dots,k\} \\
& \gamma,\, \alpha \geq 0.
\end{aligned}
\tag{NL}
$$

The nonlinear part of the model is due to the product between $x^{(j)}$ and α. To avoid the nonlinearity the authors in [41] suggest a search for local instead of global minima by considering only restricted search directions. Instead of minimizing $x(1),...,x(k)$, α, and γ simultaneously, they minimize either only $x^{(1)},\dots,x^{(k)}$, γ and keep α fix, or they minimize α and γ, and keep $x^{(1)},\dots,x^{(k)}$ fix. The first optimization problem is called x-step, the second α-step.

To solve x-step, we solve the following MILP

$$
\begin{aligned}
\min \quad & \sum_{i\in\{1,\dots,w\}} b_i\gamma_i \\
s.t. \quad & \sum_{i\in\{1,\dots,w\}} A_{it}\gamma_i \geq \sum_{j\in\{1,\dots,k\}} x_t^{(j)}\alpha_j \quad \forall\, t\in\{1,\dots,n\} \\
& x^{(j)} \in X \qquad\qquad\qquad\qquad \forall\, j\in\{1,\dots,k\} \\
& \gamma \geq 0.
\end{aligned}
\tag{x-step}
$$

To solve an α-step, we solve the following LP

$$
\begin{aligned}
\min \quad & \sum_{i\in\{1,\dots,w\}} b_i\gamma_i \\
s.t. \quad & \sum_{i\in\{1,\dots,w\}} A_{it}\gamma_i \geq \sum_{j\in\{1,\dots,k\}} x_t^{(j)}\alpha_j \quad \forall\, t\in\{1,\dots,n\} \\
& \sum_{j\in\{1,\dots,k\}} \alpha_j = 1 \\
& \gamma,\, \alpha \geq 0.
\end{aligned}
\tag{α-step}
$$

We start the local search with an x-step. As initial values for α we choose $\tilde{\alpha}_j = \frac{2j}{k(k+1)}$, $\forall\, j \in \{1,\dots,k\}$. Different values for $\tilde{\alpha}$ help to break the symmetry of the model formulation. The optimal solution of the x-step is then used to solve the first α-step. We iterate between x, and α-steps until no further improvement is found. Since the objective value decreases in each step, except of the last step, we will end up in a local minimum after a finite number of steps.

5.5 Algorithmic implementations

We solve the compact linearized formulation (LinearMaster) and the extended formulation (StrongMaster) presented earlier in this study. As described in the previous section, both formulations involved decomposition algorithms to cope with a large number of scenarios. In addition, (StrongMaster) also needs to handle a large number of solutions. We detail below the different algorithms we have implemented to handle this situation. These algorithms are based on the MILP solvers CPLEX [2] and BaPCod [1]. While CPLEX cannot handle the dynamic generation of variables, BaPCod is a toolbox specialized in branch-and-cut-and-price algorithms.

5.5.1 Compact formulation

As stated in Section 5.3, Algorithm 5 could be improved by embedding the generation of the inequalities and variables inside a single branch-cut-and-price algorithm. The solver CPLEX cannot handle the dynamic generation of variables, although, and unreported numerical results with BaPCod have shown that the compact formulation is orders of magnitude slower to solve when using a branch-and-cut-and-price algorithm instead of the row-and-column generation algorithm from Algorithm 5. We have then only implemented, for the Compact formulation, the standard row-and-column generation algorithm where scenarios added at end of MIP solution of formulation (LinearMaster). Also note that, as enumeration of solutions is not applicable, they are calculated as variables inside the formulation.

5.5.2 Extended formulation

There are two fundamentally different ways to address the formulation (StrongMaster): either we take all solutions from X from the start of the algorithm (solutions are static and denoted as PrN hereafter), or we generate these solutions on the fly in the course of a branch-and-cut-and-price algorithm (solutions are dynamic and denoted as PrY hereafter). Notice here that, unlike the new scenarios which can be added either at the end of the branch-and-bound tree or at any of its nodes, PrY needs the additional variables and constraints to be generated at each node of the branch-and-bound-tree. This is because these variables and constraints can improve the bound of any node, and omit to generate them at some nodes may lead to fathoming the nodes yielding the optimal solution.

Summarizing, we implement formulation (StrongMaster) where we take all solutions from X from the start of the algorithm (PrN), or we generate these solutions on the fly in the course of a branch-and-cut-and-price algorithm (PrY). Further specifications of these algorithms are:

Method	Formulation	Master Solver	Master Solution Generation	Separation Problem	Bounds
Com-CPX-PrN-LzN	Compact	CPLEX	Variables	Scenarios added at end of MIP solution	FALSE
Ext-CPX-PrN-LzN	Extended	CPLEX	Static	Scenarios added at end of MIP solution	FALSE
Ext-CPX-PrN-LzN-Bound	Extended	CPLEX	Static	Scenarios added at end of MIP solution	TRUE
Ext-BPC-PrN-LzN	Extended	BaPCod	Static	Scenarios added at end of MIP solution	FALSE
Ext-BPC-PrY-LzY	Extended	BaPCod	Dynamic	Scenarios added with lazy constraints	FALSE
Ext-BPC-PrY-LzY-Bound	Extended	BaPCod	Dynamic	Scenarios added with lazy constraints	TRUE
Han-CPX-PrN-LzN	Inner Dual	CPLEX	Not Applicable	Not Applicable	Not Applicable

Table 5.1: Variants for the decomposition algorithm and inner dualization algorithm

Lazy constraints Scenarios are generated dynamically within the subproblems, generating new variables and constraints. These variables and constraints can be added at the end of each iteration, after complete resolution of the mixed integer master problem (denoted as LzN hereafter), or can be added at each integer node of the branch-and-bound algorithm to solve the master problem, using the optimization feature of lazy constraints (denoted as LzY hereafter).

Treatment of the root node We generate bounds and scenarios at the root node (denoted as Bound hereafter). We solve a relaxed problem to generate as much solutions and scenarios as possible to feed our extended formulation algorithm. As described in Section 5.4, we use a heuristic solution of as an initial incumbent solution (upper bound).

Table 5.1 presents different possible combinations of algorithmic variants we have implemented and tested.

5.6 Computational experiments

We compare the performances between the different algorithmic implementations presented in Section 5.5 using random generated knapsack problem instances. For performance comparison we also solve problem (M^3) based on inner dualization mixed-integer linear reformulation for uncertain objective functions for non empty bounded polyhedron uncertainty sets presented in [69]. We name this method as Han-CPX-PrN-LzN in Table 5.1.

Since time performance of the method Com-CPX-PrN-LzN was much worse when compared to others and the majority of the problems solved with it overrun time limit, we have omitted to present their individual results.

We have created problem instances based on 3 cases of 25 jobs. For polyhedral uncertainty set we build convex budgeted uncertainty sets as defined by see [22]. We have run them for different values of $\Gamma \in \{3, 6\}$, $k \in \{2, 4, 5\}$, with the 6 algorithms presented in Table 5.1, giving a total of 108 problem instances that were run. The weights w_i were chosen randomly from set $\{100, \ldots, 1500\}$ and the knapsack capacity b was set to $100n$. We assume to have negative costs for the items to stay in the framework of minimization problems. Each knapsack instance has been equipped

with convex budgeted uncertainty sets where the mean vector $\hat{c}$ was chosen randomly with $\hat{c}_i \in \{10000, \ldots, 15000\}$ and d_i was set to $0.1\hat{c}_i$ for all $i \in \{1, \ldots, n\}$. Time limit was set to 9600s.

5.6.1 Results

We first compare time performance in Figure 5.1 using a performance profile (see [58]). In this figure, the vertical axis points out in percentage, for each algorithm, in how many instances the result was not more than x times - horizontal axis - worse than the best algorithm.

As already pointed out, the extended formulation, with its tighter relaxation bound, outperforms the equivalent problem implemented with compact formulation in any of its different algorithmic implementations. By analyzing Figure 5.1 it also becomes clear that the Extended formulation also outperforms the formulation reflected in method Han-CPX-PrN-LzN, in all variants.

The variants of lazy constraints, pricing subproblems and bound have satisfactorily improved performance of algorithms. In particular, we can see that although BaPCod cannot outperform CPLEX on basic variants (Ext-BPC-PrN-LzN versus Ext-CPX-PrN-LzN), by using its unique column generation features greatly improves its performance. In fact, Ext-BPC-PrY-LzY-Bound outperforms all other methods.

The introduction of the bound variant, with initial solutions and scenarios and lower and upper bounds, was beneficial both for CPLEX and BaPCod. The CPLEX variant Ext-CPX-PrN-LzN-Bound outperformed all other variants, except Ext-BPC-PrY-LzY-Bound.

We present in Table 5.2 detailed results for one of the cases (# 3) of 25 jobs. The total time, already used in the performance profile of Figure 5.1, includes time spent, when applicable, to generate solutions, to solve the relaxed and the heuristic problem and to solve the MIP extended or inner dual formulation. The results reinforces the beneficial effect of the introduction of lazy constraints and pricing subproblems, and afterwards the bound procedure.

Mip node count expresses the total number of nodes visited, among all iterations of the branch-and-bound algorithm used to solve the Mip problem. It can be verified the reduction of total time is linked with the reduction of the number of nodes visited.

Mip # lazy constraints and Mip # pricing columns expresses the total number of lazy constraints and pricing columns inserted, among all iterations of the branch-and-bound algorithm used to solve the Mip problem. It can be verified that the introduction of the bound variant, as expected, reduces the quantity of lazy constraints and pricing columns sub problems necessary to solve the MIP problem.

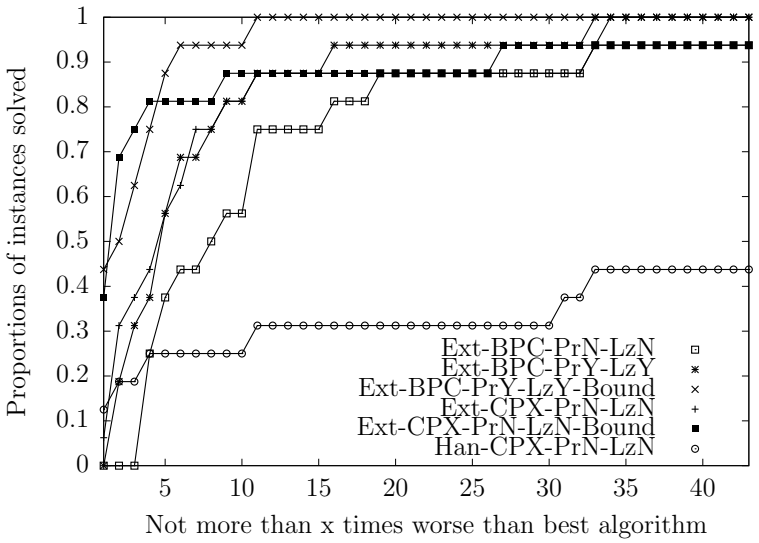

Figure 5.1: Performance among different algorithmic implementations

We register the solution of the static robust problem, the relaxed problem, the heuristic solution, the MIP solution and their corresponding gap to the optimal MIP solution as price of robustness, relaxed gap, heuristic gap and optimal gap, respectively.

The price of robustness result evidences that there is a positive gain, an average gain of 3.21% for the instances presented, in reducing conservatism of the robust solution while using the min-max-min approach.

The relaxed gap result shows that the extended formulation indeed provides a tight relaxation gap that benefits the branch-and-bound procedure used to solve the MIP problem. This property is not shared by the formulation solved in the method Han-CPX-PrN-LzN method.

The heuristic gap result shows that the algorithm used to calculate a heuristic provides a very good solution. For the instances presented in Table 5.2 the average heuristic gap is 0.41%. That positively contributes for the effectiveness of the algorithmic methods with the bound variant.

The optimal gap result shows that, for the instances not resolved within time limit, a maximum of 1.99% of optimality gap was achieved. In particular, the Han-CPX-PrN-LzN method, although not able to solve the majority of the instances presented within the time limit, was able to achieve a 0% of optimality gap in all these instances.

We also register the quantity of solutions generated by an intelligent enumeration of feasible solutions in a first phase of our algorithm. It can be verified that this enumeration is successful in drastically reducing the number of possible solutions.

Problem Instance (#,Γ,k)	Method	Static Robust Solution	Quantity Generated Solutions	Relaxed Solution	Heuristic Solution	MIP #Lazy Constraints	MIP #Pricing columns	MIP Node count	MIP Solution	Total Time	Price of Robustness	Relaxed Solution gap	Heuristic Solution gap	Optimal gap
3/3/2	Ext-BPC-PrN-LzN	5323	124	0.00	0.00	34	0	12070	5486.83	262.41	2.99%	-%	-%	0.00%
3/3/2	Ext-BPC-PrY-LzY	5323	124		0.00	33	70	951	5486.83	73.11	2.99%	-%	-%	0.00%
3/3/2	Ext-BPC-PrY-LzY-Bound	5323	124	5628.34	5429.42	21	66	823	5486.83	99.02	2.99%	2.58%	1.05%	0.00%
3/3/2	Ext-CPX-PrN-LzN	5323	124	0.00	0.00	30	0	7050	5486.83	42.24	2.99%	-%	-%	0.00%
3/3/2	Ext-CPX-PrN-LzN-Bound	5323	124	5628.34	5429.42	20	0	1289	5486.83	465.05	2.99%	2.58%	1.05%	0.00%
3/3/2	Han-CPX-PrN-LzN	5323	0	9984.10	0.00	0	0	111956	5486.83	17.42	2.99%	81.96%	-%	0.00%
3/3/4	Ext-BPC-PrN-LzN	5323	124	0.00	0.00	99	0	190235	5654.55	9600.00	5.86%	-%	-%	0.72%
3/3/4	Ext-BPC-PrY-LzY	5323	124		0.00	206	91	10000	5614.28	9600.00	5.19%	-%	-%	0.00%
3/3/4	Ext-BPC-PrY-LzY-Bound	5323	124	5628.34	5614.28	43	62	2005	5614.28	297.07	5.19%	0.25%	0.00%	0.00%
3/3/4	Ext-CPX-PrN-LzN	5323	124	0.00	0.00	148	0	326718	5633.00	9600.00	5.50%	-%	-%	0.33%
3/3/4	Ext-CPX-PrN-LzN-Bound	5323	124	5628.34	5614.28	28	0	16952	5614.28	420.61	5.19%	0.25%	0.00%	0.00%
3/3/4	Han-CPX-PrN-LzN	5323	0	16358.87	0.00	0	0	33973517	5614.28	9600.00	5.19%	191.38%	-%	0.00%
3/3/5	Ext-BPC-PrN-LzN	5323	124	0.00	0.00	91	0	189141	5677.37	9600.00	6.24%	-%	-%	0.94%
3/3/5	Ext-BPC-PrY-LzY	5323	124		0.00	262	91	10000	5624.66	9600.00	5.36%	-%	-%	0.00%
3/3/5	Ext-BPC-PrY-LzY-Bound	5323	124	5628.34	5615.42	98	67	6969	5624.66	1362.20	5.36%	0.07%	0.16%	0.00%
3/3/5	Ext-CPX-PrN-LzN	5323	124	0.00	0.00	155	0	276945	5649.51	9600.00	5.78%	-%	-%	0.44%
3/3/5	Ext-CPX-PrN-LzN-Bound	5323	124	5628.34	5615.42	51	0	32190	5624.66	930.98	5.36%	0.07%	0.16%	0.00%
3/3/5	Han-CPX-PrN-LzN	5323	0	19262.28	0.00	0	0	28099062	5624.66	9600.00	5.36%	242.46%	-%	0.00%
3/6/2	Ext-BPC-PrN-LzN	5084	252	0.00	0.00	50	0	52100	5090.96	3121.08	0.14%	-%	-%	0.00%
3/6/2	Ext-BPC-PrY-LzY	5084	252		0.00	45	89	1365	5090.96	190.21	0.14%	-%	-%	0.00%
3/6/2	Ext-BPC-PrY-LzY-Bound	5084	252	5218.70	5088.00	30	59	651	5090.96	129.52	0.14%	2.51%	0.06%	0.00%
3/6/2	Ext-CPX-PrN-LzN	5084	252	0.00	0.00	46	0	22091	5090.96	420.04	0.14%	-%	-%	0.00%
3/6/2	Ext-CPX-PrN-LzN-Bound	5084	252	5218.70	5088.00	30	0	4393	5090.96	2704.14	0.14%	2.51%	0.06%	0.00%
3/6/2	Han-CPX-PrN-LzN	5084	0	9300.47	0.00	0	0	65932	5090.96	12.55	0.14%	82.69%	-%	0.00%
3/6/4	Ext-BPC-PrN-LzN	5084	252	0.00	0.00	63	0	131319	5291.00	9600.00	3.91%	-%	-%	1.99%
3/6/4	Ext-BPC-PrY-LzY	5084	252		0.00	196	112	10000	5187.52	6022.84	2.00%	-%	-%	0.00%
3/6/4	Ext-BPC-PrY-LzY-Bound	5084	252	5218.70	5145.66	81	81	10000	5187.52	2642.51	2.00%	0.60%	0.81%	0.00%
3/6/4	Ext-CPX-PrN-LzN	5084	252	0.00	0.00	101	0	239288	5228.88	9600.00	2.77%	-%	-%	0.80%
3/6/4	Ext-CPX-PrN-LzN-Bound	5084	252	5218.70	5145.66	71	0	191113	5191.50	9600.00	2.07%	0.52%	0.88%	0.08%
3/6/4	Han-CPX-PrN-LzN	5084	0	15074.22	0.00	0	0	33736781	5187.52	9600.00	2.00%	190.59%	-%	0.00%
3/6/5	Ext-BPC-PrN-LzN	5084	252	0.00	0.00	54	0	97416	5293.63	9600.00	3.96%	-%	-%	1.71%
3/6/5	Ext-BPC-PrY-LzY	5084	252		0.00	185	94	10000	5204.89	9600.00	2.32%	-%	-%	0.00%
3/6/5	Ext-BPC-PrY-LzY-Bound	5084	252	5218.70	5187.52	63	74	10000	5204.89	9600.00	2.32%	0.27%	0.33%	0.00%
3/6/5	Ext-CPX-PrN-LzN	5084	252	0.00	0.00	112	0	279285	5247.04	9600.00	3.11%	-%	-%	0.81%
3/6/5	Ext-CPX-PrN-LzN-Bound	5084	252	5218.70	5187.52	53	0	98669	5204.89	4070.34	2.32%	0.27%	0.33%	0.00%
3/6/5	Han-CPX-PrN-LzN	5084	0	17548.43	0.00	0	0	24784382	5204.89	9600.00	2.32%	237.15%	-%	0.00%

RCG meaning row-and-column generation

Table 5.2: Detailed results for all methods and instance # 3

Chapter 6

Distributionally robust fleet assignment problem

6.1 Introduction

To provide a high-quality, low-cost product, airlines rely on optimization-based decision support systems to generate profitable and cost-effective fare classes, flight schedules, fleet plans, aircraft routes, crew pairings, gate assignments, maintenance schedules, food service plans, training schedules, and baggage handling procedures [127].

Once an airline decides when and where to fly (flight legs) by developing its flight schedule, the next decision is determining the type of aircraft, or fleet, that should be used on each of the flight legs defined within the flight schedule. This process is called fleet assignment and its purpose is to assign fleet types to flight legs, subject to an available number of aircrafts and conservation of aircraft flow requirements, such as to maximize profits with respect to captured passenger demand. This means the fleet assigment has a direct effect on the airline revenue and operating costs and thus is considered to have significant impact on airline profits. This decision needs to be made well in advance of departures when passenger demand is still highly uncertain. The factors that influence schedulers when assigning fleet types to various flights are: passenger demand, seating capacity, operational costs, and availability of maintenance at arrival and departure stations. One important requirement of the fleet assignment is that the aircraft must circulate in the network of flights. These so-called balance constraints are enforced by using time lines to model the activities of each fleet type. The period for which the assignment is done is normally one day for domestic flights.

Profit maximization is normally defined in terms of unconstrained revenue minus assignment cost. Unconstrained revenue of a flight leg is the maximum attainable

revenue for that particular flight regardless of assigned capacity. Assignment cost, a function of the assigned fleet type, includes the flight operating cost, passenger carrying related cost and spill cost. Spill cost on a flight is the revenue lost when the assigned aircraft for that flight cannot accommodate every passenger. The result is that either the airline spills some passengers to other flights in its own network (in which case these passengers are recaptured by the airline), or they are spilled to other airlines.

In [11], the authors point out that, in the airline industry, despite the best efforts to optimize and execute a plan, unforeseen disruptions to this plan occur inevitably on every single day. These result in what the authors call irregular operations. One way to mitigate the effects of irregular operations is to focus on optimization-based recovery techniques – replanning aircraft, crew and passenger routings. An alternative approach that aims at preparing robust airline operations is robust planning, that is explicitly aimed at reducing the complexity and/or need for recovery by generating, during the planning process, plans that are resilient to disruptions. This allows planned operations to be more consistent, or require less extensive adjustments to maintain operability.

In particular, as passenger demand is uncertain, in some works the authors have assumed the random nature of demand uncertainty by using stochastic programming modeling techniques.

In [95] the authors present a two-stage stochastic programming model for fleet assignment. At the first stage, the model assigns aircraft families to flight legs. In a second stage, the model assigns aircraft types to flight legs. Passenger demand and aircraft capacity are implicitly taken into account in the model through the revenue and the operating cost associated with each scenario. The estimate of the expected revenue is determined as the average over the scenarios. The main contribution of this paper is a method to obtain an approximation for the expected profit function using a regression splines fit.

In [111] the authors also formulate the airline fleet assignment as a two-stage stochastic programming model. They consider passenger demand uncertainty by the use of demand scenarios and develop a Benders decomposition-based approach to facilitate handling of large scale problems.

In [75] the authors develop a two-stage stochastic programming model for integrated flight scheduling and fleet assignment where the fleet family assigned to each scheduled flight leg is decided at the first-stage. Then, the fleet type to assign to each flight leg is decided at the second-stage based on demand and fare realization. Sample average approximation (SAA) algorithm is then used to solve the problem and provide information on the quality of the solution.

In [36] the authors propose a new model based on itinerary grouping to mitigate

the effect of demand uncertainty. Their itinerary group fleet assignment model deals with the difficulties caused by itinerary forecast by replacing them with aggregated demand forecasts. The authors affirm that an itinerary-based representation of demand (see Section 6.2 for details) has led to a high granularity of demand, making it hard to predict.

In this work, as an alternative to previous works presented, we propose a two-stage data-driven distributionally robust optimization model to address the question of airline robust planning for the fleet assignment problem. This novell modeling approach comprises our main contribution with respect to the airline fleet assignment problem literature.

Our model considers the stochastic nature of uncertainty of passenger demand and follows the guidelines that one should generate, during the planning process, plans that are resilient to disruptions.

We adopt the concept of robust optimization as defined in [17] and [18] in that the demand uncertainty belongs to a known deterministic uncertainty set. In fact, we consider this uncertainty set as the support for the family of probability distributions associated with our random passenger demand parameter. We consider a data-driven approach by which this uncertainty set is constructed from available historical data. We assume that historical unconstrained (not subject to capacity issues) itinerary demand data is available and that we can use this historical data to predict future demand. By constructing the uncertainty set from historical data we are able to capture correlations between demands of different itineraries and thus mitigate the granularity demand effect as pointed out in [36].

On the other hand, we consider different modeling alternatives to mitigate conservatism of a robust approach. Since fleet assignment is a repetitive process, where fleet assignment decisions are made on daily basis, we mitigate the conservatism of the worst-case objective of classical robust optimization and consider a distributionally robust optimization approach on which we optimize the worst case expected performance on a set constituted by an infinite number of probability distributions, named ambiguity set (see [53] for main concepts). We also propose a two-stage model, as introduced in [19] where, although all the fleet assignments decisions are first stage, the calculation of lost revenue (spill) is only done after realization of uncertainty.

To facilitate handling large-scale fleet assignment problems, we propose the use of principal component analysis techniques to reduce dimension of the uncertainty set and the use of affine decision rules for our two-stage problem as approximations to improve time performance of our algorithms.

The remainder of this chapter is organized as follows. In Section 6.2 we review existing formulations for the fleet assignment problem and propose our two-stage dis-

tributionally robust formulation. Next, in Section 6.3 we reformulate our problem based on assumptions we make on the ambiguity set and the use of affine decision rules to approximate our problem. Section 6.4 details our ambiguity set and the algorithm developed to construct it using machine learning techniques. There we present also how principal component analysis techniques can be used to reduce dimension of the uncertainty set and approximate the problem. In Section 6.5 we present computational results using a hub-and-spoke airline instance example created for our testing purposes. There we compare solutions provided by our formulation to other robust formulations and a deterministic fleet assignment formulation where a scenario-based passenger demand vector is considered as input. We compare solutions by simulating an airline operation for a predetermined period of time.

6.2 Fleet assignment formulations

The fleet assignment model is typically formulated as a mixed-integer program. One can see the work in [112] for a survey of different modeling approaches for the problem.

In [3] the author first introduced for the fleet assignment problem a time-space network model to represent the availability of the fleet at each airport in the course of time. The proposed model resulted in a linear program that could either maximize profit or minimize operations cost.

In [70] the authors use the time-space network model and develop a large-scale integer program for fleet assignment. They propose several preprocessing techniques, namely node aggregation and isolated islands at stations, in order to reduce problem complexity.

In these two works demand is expressed for a specific flight leg and, therefore, these works do not capture demand dependencies between legs. This is because demand is defined for airline itineraries that can be comprised by multiple flight legs. Variations of demand in one itinerary flight leg will affect the others legs. This is called network effect and it was taken into consideration in the model defined in [10]. There, the authors use the time-space network model and consider the effect of recapturing, where passengers spills from one itinerary can be redirected to alternative itineraries. In their model demand is deterministic.

This model is reference for our work, but we do not consider recapturing. We replicate here the itinerary based formulation as presented in [36] where the authors also explicitly deal with itinerary fare classes to better capture the revenue dimension by favoring higher classes instead of considering all the fare classes at the same level. We present notations and formulation used.

Sets

P: the set of itinerary fare classes, indexed by p

A: the set of airports, indexed by o

L: the set of flight legs, indexed by i

K: the set of fleet types, indexed by k

T: the sorted set of all relevant event times (leg departures or aircraft availability) at all airports, indexed by t

$CL(k)$: the set of flight legs that pass the count time when flown by fleet type k

$I(k, o, t)$: the set of inbound flight legs to node (k, o, t)

$O(k, o, t)$: the set of outbound flight legs from node (k, o, t)

Decision variables

t_p: the number of passengers requesting itinerary fare class p and spilled by the model because of the capacity limit.

f_{ki}: binary variable equal to 1 if fleet type k is assigned to flight leg i, 0 otherwise.

y_{kot^+}: the number of fleet type k that are on the ground at airport o immediately after time t.

y_{kot^-}: the number of fleet type k that are on the ground at airport o immediately before time t. If $t^{'}$ is the time of the first event occurring after t, then $y_{kot} = y_{kot'^-}$

Data

$SEATS_k$: the number of seats available on aircraft of fleet type k.

c_{ki}: the cost of operating leg i with fleet type k.

N_k: the number of aircraft in fleet type k.

D_p: the unconstrained demand for itinerary fare class p.

$fare_p$: the fare class for itinerary p.

δ_i^p: a binary flag equal to 1 if itinerary fare class includes flight leg i, 0 otherwise.

count time: the time at which a snapshot of fleet utilization is taken to ensure consistency with the available fleet.

t_m: the last event before the *count time*, $t_m = count\,time^-$.

$$
\begin{aligned}
(IFAM) \quad \min \quad & \sum_{i\in L, k\in K} c_{ki}\, f_{ki} + \sum_{p\in P} fare_p\, t_p && && (6.1)\\
\text{s.t.} \quad & \sum_{k\in K} f_{ki} = 1 && \forall i \in L && (6.2)\\
& \sum_{i\in I(k,o,t)} f_{ki} + y_{kot^-} = \sum_{i\in O(k,o,t)} f_{ki} + y_{kot^+} && \forall k, o, t && (6.3)\\
& \sum_{o\in A} y_{kot_m} + \sum_{i\in CL(k)} f_{ki} \le N_k && \forall k \in K && (6.4)\\
& \sum_{p\in P} \delta_i^p D_p - \sum_{p\in P} \delta_i^p t^p \le \sum_{k\in K} f_{ki}\, SEATS_k && \forall i \in L && (6.5)\\
& t_p \le D_p && \forall p \in P && (6.6)
\end{aligned}
$$

The objective function (6.1) minimizes the total cost of operations plus the cost related to spilled itinerary fare class demand. This minimization is equivalent to profit maximization. Constraints (6.2) are the leg coverage constraints. Each flight leg has to be operated by exactly one aircraft type. The flow conservation constraint related to each single event is ensured through constraints (6.3). The limited size of each fleet is respected through constraints (6.4). The count time can be seen as a fixed time where a cut is applied on the network to ensure that the total aircraft of each fleet type k on the ground at all airports plus the one flying at the time must not exceed the total aircraft N_k available for type k. The capacity constraints (6.5) ensure that satisfied demand fits with the number of seats available on any given leg. Last, constraints (6.6) ensure that spill does not exceed unconstrained demand for any given itinerary fare class.

We now propose a two-stage distributionally robust optimization formulation derived from $IFAM$ formulation to incorporate the random nature of passenger demand vector D. Distributionally robust optimization is an emerging and effective method to address the inexactness of probability distributions of uncertain parameters.

As an extension to robust optimization that only considers the worst-case parameter realization in the uncertainty set, without necessarily considering random nature of uncertainty, distributionally robust optimization hedges against the worst-case probability distribution in the ambiguity set, as described in Section 6.1. The worst-case realization that robust optimization hedges against might be unrealistic in practice. The ambiguity set is typically constructed based on partial distributional information, such as support set and moment statistics, which can be obtained from available historical data.

We formulate our problem assuming that passenger demand spill decision variable is a second stage variable. This way passenger demand spill is only defined after realization of uncertainty and we represent this dependency defining it as a function map $t_p(D)$. We also assume the uncertainty of vector D is represented through a probability distribution $\mathbb{P}$ that belongs to an ambiguity set $\mathcal{D}$.

We present the formulation developed.

$$
\begin{aligned}
(DIFAM) \quad \min \quad & \sum_{i\in L,k\in K} c_{ki}\, f_{ki} + \sup_{\mathbb{P}\in\mathcal{D}} \mathbb{E}_{\mathbb{P}}[Q(f,D)] && && (6.7)\\
\text{s.t.} \quad & \sum_{k\in K} f_{ki} = 1 && \forall i \in L && (6.8)\\
& \sum_{i\in I(k,o,t)} f_{ki} + y_{kot^-} = \sum_{i\in O(k,o,t)} f_{ki} + y_{kot^+} && \forall k,o,t && (6.9)\\
& \sum_{o\in A} y_{kot_m} + \sum_{i\in CL(k)} f_{ki} \le N_k && \forall k \in K && (6.10)\\
& && && (6.11)
\end{aligned}
$$

where

$$
\begin{aligned}
Q(f,D) = \quad \min \quad & \sum_{p\in P} fare_p\, t_p(D) && && (6.12)\\
& \sum_{p\in P} \delta_i^p D_p - \sum_{p\in P} \delta_i^p t^p(D) \le \sum_{k\in K} f_{ki}\, SEATS_k && \forall i \in L && (6.13)\\
& t_p(D) \le D_p && \forall p \in P && (6.14)
\end{aligned}
$$

The cost $\sum_{i\in L,k\in K} c_{ki}\, f_{ki}$ incurred during the first stage is deterministic. In progressing to the second stage, the random passenger demand vector D is realized. We can then determine the cost incurred at the second stage. For a given first stage fleet type assignment decision, f, and a realization of the random passenger demand vector, D, we evaluate the second stage cost via the linear optimization problem, $Q(f,D)$. Since the fleet type assignment is a repetitive daily process and the true probability distribution of D is unknown and belong to a family of distributions set $\mathcal{D}$ we are interested in the worst case expectation $\sup_{\mathbb{P}\in\mathcal{D}} \mathbb{E}_{\mathbb{P}}[Q(f,D)]$. Note that it is a relatively complete recourse problem because any first-stage solution leads to a feasible second-stage solution.

In order to be able to deal with large scale problems, our two-stage distributionally robust optimization formulation must admit a tractable reformulation. The reformulation is closely related to the choices of ambiguity set that we make. On the

other hand these choices must correctly reflect properties of historical data available.

In the next section we show that defining ambiguity sets by linear relationships of uncertainty parameters and approximating second stage variables as affine functions of uncertainty parameters derives a tractable problem. We will also use techniques of uncertainty dimensionality reduction as a further compromise between optimality and tractability.

6.3 Two-stage distributional reformulation

6.3.1 Dimensionality reduction

In real case examples, the dimension of the random passenger demand vector D can be in the range of thousands of itineraries. This can impact performance of the formulation $DIFAM$. Employing dimensionality reduction techniques to reduce the number of random variables under consideration can improve performance of our formulations.

A linear technique for dimensionality reduction, principal component analysis, performs a mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized. In practice, the covariance matrix of the data is constructed and the eigenvectors of this matrix are computed. The eigenvectors that correspond to the largest eigenvalues, that are called the principal components vectors, can now be used to reconstruct a large fraction of the variance of the original data. The original space of itineraries is reduced afterwards with data loss, but hopefully retaining the most important variance. See [123] for more details on principal component analysis (PCA).

We assume there is a set $\mathcal{W}'$ of N demand data samples available, $\mathcal{W}' = \{D^{(i)}\}_{i=1}^{N}$, based on historical data, and use this set to calculate the mean vector $\bar{D}$ and variance vector $\hat{D}$. We execute a procedure to decompose the passenger demand vectors as components, X_c, of the principal components vectors, Y^c, where $c \in \{1, \ldots, C\}$ are the indexes of the selected principal components vectors and the principal component vector has dimension $|P|$.

Before decomposing we normalize each instance of the demand vector, $D^{(i)}$, using $\bar{D}$ and $\hat{D}$. We then compute the coordinates, X_c, in the system of coordinates of the principal components vectors, Y^c. This is given by the expression

$$D_p^{(i)} = \bar{D}_p + \hat{D}_p \sum_{c=1}^{C} X_c^{(i)} Y_p^c \quad \forall p \in P. \tag{6.15}$$

In the following we need the random vector to be non negative. Hence, we introduce a new random vector ξ where each component will vary in the nonnegative

interval $[0, 1]$. We do this by defining the components of each demand vector $D^{(i)}$ as

$$D_p^{(i)} = D1_p + \sum_{c=1}^{C} D2_{pc}\xi_c^{(i)}, \tag{6.16}$$

where

$$D1_p = \bar{D}_p + \hat{D}_p \sum_{c=1}^{C} \min_i (X_c^{(i)}) Y_p^c, \tag{6.17}$$

$$D2_{pc} = \hat{D}_p (\max_i (X_c^{(i)}) - \min_i (X_c^{(i)})) Y_p^c, \tag{6.18}$$

and $\max_i (X_c^{(i)})$, $\min_i (X_c^{(i)})$ are, respectively, the maximum and minimum decomposition values along each vector Y^c considering all instances, $i \in \{1, \ldots, N\}$.

This is important step in order to guarantee positive definite matrices in the algorithm developed in Section 6.4 for our distributionally robust ambiguity set.

6.3.2 Ambiguity set and first-order deviation moment functions

The tractability of a distributionally robust linear optimization problem is dependent on the choice of the ambiguity set. Several ambiguity sets have been proposed in the literature. In particular, moment-based uncertainty sets assume that all distributions in the distribution family share the same moment information. By leveraging conic duality many distributionally robust optimization problems with moment-based ambiguity sets can, in general, be reformulated equivalently as convex problems. Although these problems can be solved theoretically in polynomial time, they are not efficient for large-scale instances.

In [35] the authors propose an alternative with moment-based second-order conic (SOC) representable ambiguity sets. They show that the adaptive distributionally robust linear optimization problem can be formulated as a classical robust optimization problem.

In [35], a moment-based second-order conic representable ambiguity set, $\mathcal{D}$, is defined as

$$\mathcal{D} = \left\{ \mathbb{P} \in \mathcal{P}_0(\mathbb{R}^{|P|}) \;\middle|\; \begin{array}{c} D \in \mathbb{R}^{|P|} \\ \mathbb{E}_{\mathbb{P}}[GD] = \mu \\ \mathbb{E}_{\mathbb{P}}[g_i(D)] \leq \gamma_i \quad \forall i \in I \\ \mathbb{P}(D \in U) = 1 \end{array} \right\}.$$

We assume random passenger demand vector D, but the same results can be

derived substituting for dimensional reduced random vector ξ. $\mathcal{P}_0(\mathbb{R}^{|P|})$ represents the set of all probability distributions in $\mathbb{R}^{|P|}$ and new parameters are defined as $G \in \mathbb{R}^{n_1 \times |P|}$, $\mu \in \mathbb{R}^{n_1}$, $\gamma \in \mathbb{R}^{|I|}$, SOC representable support set $U \in \mathbb{R}^{|P|}$ and SOC representable functions $g_i \in \mathbb{R}^{|P| \times 1}$.

$\mathcal{D}$ only contains valid distributions supported over the support set U and moment information of uncertainties are characterized via functions g_i. The equality expectation expression allow the modeler to specify the mean values of D.

The authors of [35] further reformulate the ambiguity set $\mathcal{D}$ as a projection of an extended ambiguity set $\bar{\mathcal{D}}$ by introducing an I-dimensional auxiliary random vector u in

$$\bar{\mathcal{D}} = \left\{ \mathbb{Q} \in \mathcal{P}_0(\mathbb{R}^{|P|} \times \mathbb{R}^{|I|}) \;\middle|\; \begin{array}{c} (D,u) \in \mathbb{R}^p \times \mathbb{R}^{|I|} \\ \mathbb{E}_{\mathbb{Q}}[GD] = \mu \\ \mathbb{E}_{\mathbb{Q}}[u] \leq \gamma_i \quad \forall i \in I \\ \mathbb{P}((D,u) \in \bar{U}) = 1 \end{array} \right\}$$

where $\bar{U}$ is the lifted support set defined as

$$\bar{U} = \left\{ (D,u) \in \mathbb{R}^{|P|} \times \mathbb{R}^{|I|} \;\middle|\; \begin{array}{c} D \in U \\ g_i(D) \leq u_i \quad \forall i \in I \end{array} \right\}$$

They observe that the lifted ambiguity set has only linear expectation constraints and show that the adaptive distributionally robust optimization problem can be reformulated as a classical robust optimization problem with uncertainty set $\bar{U}$.

To be able to reformulate adequately our fleet assignment problem $DIFAM$ we must then define an ambiguity set $\mathcal{D}$ that will lead to a polyhedral lifted support set $\bar{U}$.

In [107] the authors define first-order deviation moment-based functions $g_i(.)$ that are second-order conic representable as piecewise linear functions

$$g_i(D) = \max\{h_i^T D - q_i, 0\} \quad \forall i \in I.$$

They can be understood as the first-order deviation of uncertain parameters along a certain projection h_i truncated at q_i. We apply these moment-based functions to our problem and also assume that the support set U is a polyhedral. We

then define our ambiguity set $\mathcal{D}$ as

$$\mathcal{D} = \left\{ \mathbb{P} \in \mathcal{P}_0(\mathbb{R}^{|P|}) \;\middle|\; \begin{array}{c} D \in \mathbb{R}^{|P|} \\ \mathbb{E}_{\mathbb{P}}[\max\{h_i^T D - q_i, 0\}] \leq \gamma_i \quad \forall i \in I \\ \mathbb{P}(D \in U) = 1 \end{array} \right\}$$

and the lifted support set $\bar{U}$ will be a polyhedral given as

$$\bar{U} = \left\{ (D,u) \in \mathbb{R}^{|P|} \times \mathbb{R}^{|I|} \;\middle|\; \begin{array}{c} D \in U \\ 0 \leq u_i \quad \forall i \in I \\ h_i^T D - q_i \leq u_i \quad \forall i \in I \end{array} \right\}$$

6.3.3 Affine decision rules

With the above definition of lifted support set we can apply, to our $DIFAM$ formulation, the reformulation proposed by [35] for the adaptive distributionally robust optimization problem, approximating second stage variables t_p as affine functions of the lifted support set parameters (D,u), $t_p(D,u) = t_p^0 + \sum_{i \in P} t_{pi}^1 D_i + \sum_{i \in I} t_{pi}^2 u_i$.

This reformulation is based on the dualization of the inner problem of $DIFAM$, $\sup_{\mathbb{P} \in \mathcal{D}} \mathbb{E}_{\mathbb{P}}[Q(f,D)]$, and by introducing Lagrangian multipliers r and β to it (alternatively see [107] for a summarized proof of this reformulation). This leads to the following classical robust optimization problem:

$$
\begin{aligned}
(RRIFAM) \quad \min \quad & \sum_{i\in L, k\in K} c_{ki}\, f_{ki} + r + \sum_{i\in I} \gamma_i \beta_i & \\
\text{s.t.} \quad & \beta_i \geq 0 & \forall i \in I \\
& r + \sum u_i \beta_i \geq \sum_{p\in P} fare_p t_p(D,u) & \forall (D,u) \in \bar{U} \\
& \sum_{p\in P} \delta_i^p D_p - \sum_{p\in P} \delta_i^p t^p(D,u) \leq \sum_{k\in K} f_{ki}\, SEATS_k & \forall i \in L,\, \forall (D,u) \in \bar{U} \\
& t_p(D,u) \leq D_p & \forall p \in P,\, \forall (D,u) \in \bar{U} \\
& \sum_{k\in K} f_{ki} = 1 & \forall i \in L \\
& \sum_{i\in I(k,o,t)} f_{ki} + y_{kot^-} = \sum_{i\in O(k,o,t)} f_{ki} + y_{kot^+} & \forall k, o, t \\
& \sum_{o\in A} y_{kot_m} + \sum_{i\in CL(k)} f_{ki} \leq N_k & \forall k \in K \\
& t_p(D,u) = t_p^0 + \sum_{i\in P} t_{pi}^1 D_i + \sum_{i\in I} t_{pi}^2 u_i & \forall p \in P
\end{aligned}
$$

6.4 Data-driven ambiguity set

A desirable ambiguity set should flexibly adapt to the intrinsic structure behind real data, thereby well characterizing $\mathbb{P}$ and attempting to reduce natural conservatism of robust solutions. In face of complicated distributional geometry, making prior assumptions on the form of probability distribution or using classical uncertainty sets to describe their support have limited modeling power. With this in mind we adopt a data-driven methodology to construct and define parameters of the support set and moment-based functions associated with our ambiguity set.

6.4.1 Support Set

In what follows, we use the technical approach of [108] to construct a support set U from data samples of the random variable ξ. We assume there is a set $\mathcal{W}$ of N data samples available, $\mathcal{W} = \{\xi^{(i)}\}_{i=1}^N$, and this set is constructed from the sample of demand vectors as explained in Subsection 6.3.1.

In [108] the authors propose piecewise linear kernel-based support vector clustering (SVC) as a machine learning technique tailored to data-driven robust optimization. The distributional geometry of uncertain data can be effectively captured as a polyhedral uncertainty set, which considerably reduces conservatism of robust optimization problems. In addition, by exploiting statistical properties of SVC, the fraction of data coverage of the data-driven uncertainty set can be easily selected by

adjusting only one parameter, which furnishes an interpretable and pragmatic way to control conservatism and exclude outliers.

Support vector clustering [16] maps data points by means of a kernel to a high dimensional feature space, searching for the minimal enclosing sphere. This can be formulated as an optimization problem in

$$\begin{aligned} \min_{a,R} \quad & R^2 + \frac{1}{Nv}\sum_{i=1}^{N}\zeta_i \\ \text{s.t.} \quad & \left\|\phi(\xi^{(i)}) - a\right\|^2 \leq R^2 + \zeta_i \qquad \forall i \in \{1,\ldots,N\} \\ & \zeta_i \geq 0 \qquad \forall i \in \{1,\ldots,N\}, \end{aligned}$$

where the sphere is centered at vector a and of radius R and $\phi(\xi) : \mathbb{R}^{|P|} \mapsto \mathbb{R}^F$ is a nonlinear mapping to a high-dimensional feature space $\mathcal{F}$. A soft margin is adopted by adding slack variables ζ_i with an adjusting regularization parameter $v \geq 0$.

Introducing Lagrangian multipliers α and β and applying Karush-Kuhn-Tucker conditions (see [16]) results in a quadratic program problem as

$$\begin{aligned} \min_{\alpha} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j K(\xi^{(i)},\xi^{(j)}) - \sum_{i=1}^{N}\alpha_i K(\xi^{(i)},\xi^{(j)}) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq \frac{1}{Nv} \qquad \forall i \in N \\ & \sum_{i=1}^{N}\alpha_i = 1 \end{aligned}$$

where the kernel trick $K(\xi^{(i)},\xi^{(j)}) = \phi(\xi^{(i)})^T\phi(\xi^{(j)})$ allows for a simple computation of inner-products in feature space $\mathcal{F}$ and the center a is given by $a = \sum_{i=1}^{N}\alpha_i\phi(\xi^{(i)})$. The problem permits a geometric interpretation where

$$\begin{aligned} &\left\|\phi(\xi^{(i)}) - a\right\|^2 < R^2 \rightarrow \alpha_i = 0, \quad \beta_i = \frac{1}{Nv} \\ &\left\|\phi(\xi^{(i)}) - a\right\|^2 = R^2 \rightarrow 0 < \alpha_i < \frac{1}{Nv}, \quad 0 < \beta_i < \frac{1}{Nv} \\ &\left\|\phi(\xi^{(i)}) - a\right\|^2 > R^2 \rightarrow \alpha_i = \frac{1}{Nv}, \quad \beta_i = 0 \end{aligned}$$

Only data samples $\xi^{(i)}$ with positive α_i, referred to as support vectors, contributes to the center a. A portion of them reside exactly on the boundary of the sphere with $\left\|\phi(\xi^{(i)}) - a\right\|^2 = R^2$, named boundary support vectors, whereas the rest with $\left\|\phi(\xi^{(i)}) - a\right\|^2 > R^2$ are regarded as outliers. We hence define $SV = \{i \in \{1,\ldots,N\} \mid \alpha_i > 0\}$ and $BSV = \{i \in \{1,\ldots,N\} \mid 0 < \alpha_i < \frac{1}{Nv}\}$ as the index sets of all support vectors and boundary support vectors, respectively.

The radius R can be determined as the distance from the center a to any boundary support vector:

$$R^2 = K(\xi^{(j)}, \xi^{(j)}) - 2\sum_{i=1}^{N} \alpha_i K(\xi^{(j)}, \xi^{(i)}) + \sum_{i=1}^{N}\sum_{k=1}^{N} \alpha_i \alpha_k K(\xi^{(i)}, \xi^{(k)}), \quad j \in BSV$$

The authors in [108] then define what in our case is a data-driven support set U, as the region inside or in the borders of the sphere and is given as

$$U = \{\xi \mid K(\xi, \xi) - 2\sum_{i=1}^{N} \alpha_i K(\xi, \xi^{(i)}) + \sum_{i=1}^{N}\sum_{k=1}^{N} \alpha_i \alpha_k K(\xi^{(i)}, \xi^{(k)}) \leq R^2\}$$

Furthermore, the authors propose a weighted generalized intersection kernel that ensures positive-definiteness of the kernel matrix K, preserving the convexity of the quadratic program problem and derives a support set U that is a polyhedral, is enclosed around the origin and where each dimension of uncertain data exert similar influence on the kernel expression. Since the components of our random vector ξ are decorrelated and expressed in the same scale we suppress the weighting procedure and the kernel, that assumes non-negativity of data samples, is given by

$$K(\xi^{(i)}, \xi^{(j)}) = \sum_{k=1}^{N} l_k - \left\| \xi^{(i)} - \xi^{(j)} \right\|_1$$

where $l_k = \max_{1 \leq i \leq N} \xi_k^{(i)} - \min_{1 \leq i \leq N} \xi_k^{(i)}$.

6.4.2 Moment-based functions

For moment-based functions we adopt the work of [107] where the authors define a two-step procedure for determining parameters h_i and q_i of our piecewise linear functions in order to capture meaningful information from available data.

The directions h_i are based on principal component analysis (PCA) such that the data space becomes decorrelated along each direction and the information overlap between different directions is slight. Since our random vector ξ already comprises decorrelated components we adopt vector h_i as standard unit vectors e_i.

After that, several truncation points q_i are set along each direction h_i. For each direction h_i we choose $2J + 1$ well-distributed truncation points. The first truncation point is set as the mean value $\bar{\xi}_i$ and the remaining $2J$ ones around the mean $\bar{\xi}_i$ symmetrically based on a fixed step-size given as the variance $\hat{\xi}_i$ along the i-th direction.

In this way, we will have $C(2J + 1)$ piecewise functions $g_i(.)$ in total in the ambiguity set.

Intuitively, the parameter J can be deemed as the "size" of the ambiguity set, which can be manipulated to adjust the conservatism of the model. The more truncation points we have, the more statistical information will be incorporated, which leads to a smaller ambiguity set as well as a less conservative solution.

After determining the value of h_i and q_i, the next step is to estimate the parameters γ_i empirically based on N available data samples:

$$\gamma_i = \frac{1}{N} \sum_{j=1}^{N} \max(h_i^T \xi^{(j)} - q_i, 0)$$

Intuitively, with the values of size parameters γ_i increasing, the DRO model becomes more conservative.

6.5 Implementation and Results

6.5.1 Implementation details

We experiment the formulations proposed for the airline fleet assignment problem. Our objective is to verify the performance of each solution in a long run operation since fleet assignment is a daily repetitive process.

For our purposes, we create a small-sized hub-and-spoke airline instance, in which a unique major airport serves as a central point for coordinating flights to and from other airports. This way all our itineraries are composed of a maximum of two flight legs. We consider a structure of 9 airports, 3 fleet types and 24 daily itineraries based on three fare classes. A flight schedule with 21 flight legs is created and they are used to compose the daily itineraries.

We test this operation under four different problem formulations: $IFAM$, $RRIFAM$, as already presented in this study and two other formulations $RIFAM$ and $RIFAM2$. Formulation $RIFAM$ is a standard two-stage robust formulation where the objective is given as the worst case performance and dimensionality reduction is performed the same way as for $RRIFAM$. Formulation $RIFAM2$ is the same as $RIFAM$ where no dimensionality reduction is performed.

We randomly generate a set of 400 demand vectors using seminal uniform distributed vectors, but designing many itinerary demands as highly correlated.

We use 100 demand vectors as historical training data to create the ambiguity set of formulation $RRIFAM$ and the 300 others to simulate the airline operating period.

We use naive approaches to determine demand vectors for formulations $IFAM$, $RIFAM$ and $RIFAM2$. For formulation $IFAM$ we consider three demand scenarios of low, medium and high total demand and consider the average of these three

scenarios as input to our $IFAM$ formulated problem. For formulation $RIFAM$ and $RIFAM2$ we consider maximum and minimum demand values for each leg and consider a box uncertainty set where each demand component varies within this interval.

With the solution of formulations $IFAM$,$RIFAM$ and $RRIFAM$ we simulate an airline operating period of 300 days and calculate an objective of total operating costs plus total loss revenue. We compare simulation results of the three formulations where our focus is on analyzing objective value and time performance.

Conservatism regulation parameters of our ambiguity set are fixed as $v = 0.6$ and $J = 0$. With $v = 0.6$, 100% of demand vectors were considered inside or in the border of the support set (no outliers). We calculate parameter C so that the sum of variances in the direction of each principal component considered sums up to a minimum of 90% of the sum of variances considering all principal components. For the instance we created, $C = 8$ of 24.

To solve formulations $RIFAM$, $RIFAM2$ and $RRIFAM$ we use a master and adversarial problem approach where, at each iteration, we use the adversarial problems to search for a demand scenario instance that invalidates the master problem solution. See [34] for more details on this solution approach.

Algorithms were coded in Julia [79] using JuMP and Cplex 12.7. All algorithms were run in an Intel CORE i7 CPU 3770 machine. A limit of 9600s was given to run the algorithms.

6.5.2 Comparative performance of the formulations

Table 6.1 presents the results of the implementation and solution for the four different formulations. The objective value of $IFAM$ is 335747, of $RIFAM$ is 638817, of $RIFAM2$ is 651081 and of $RRIFAM$ is 488135. The relation between these values are as expected since formulation $IFAM$ is optimizing against a specific demand scenario, formulations $RIFAM$ and $RIFAM2$ are optimizing against a worst case scenario and formulation $RRIFAM$ is optimizing an expected performance (worst-case). $RIFAM$ is designed to be a lower bound of $RIFAM2$ since it considers less constraints (restricted uncertainty set), but the results show that $RIFAM$ is a reasonable approximation of $RIFAM2$. Since we use affine decision rules, formulations $RIFAM$, $RIFAM2$ and $RRIFAM$ are themselves upper bound approximations of the true optimal worst-case or worst-case expected performance. Since we use auxiliary variables to compose affine decision rules for formulation $RRIFAM$, it leads to more flexible results than affine decision rules use original demand uncertainty.

The total time performance result is in direct link with the number of variables of each formulation, although the number of iterations for each of the robust for-

	$IFAM$	$RIFAM$	$RIFAM2$	$RRIFAM$
Objective value	335747	638817	651081	488135
Total Time (s)	4.5	239.77	10950.47	9041.14
Number of variables	789	982	1366	1181
Number of constraints	450	-	-	-
Number of iterations	-	32	49	81
Simulation Total cost*	1.38e8	1.35e8 (2.2%)	-	1.32e8 (4.3%)

**In parenthesis percentage gain when compared to worst result*

Table 6.1: Implementation and performance comparison of fleet assignment formulations

mulations varies. In terms of time performance, dimensionality reduction has been effective to reduce total time. On the other hand, since the size of our airline instance is small, additional measures should be put in place to be able to deal with real large airline instances.

The simulation results are also as expected since the formulation $RRIFAM$, in the long run, leads to the less costly total solution. We note that there are no guarantees, in terms of the mathematical model proposed, on how formulations $IFAM$ and $RIFAM$ would perform in the long simulation run. We also note that formulation $RRIFAM$ is an approximation of the true optimal result. Even though we would expect that, in the long simulation run, result of worst-case expected performance of formulation $RRIFAM$ would out perform the two other formulations, and that is the case.

Chapter 7

Conclusions

We have reviewed recent developments in the literature of robust optimization in operations research and management science. We present new algorithmic approaches based on these developments, by applying them to different applications. Robust optimization, including its new developments, has been more and more applied to different application-specific frameworks. In particular in areas such and finance, health-care and energy systems we have seen many developments. Also it is important noticing the tendency of integration of robust optimization with stochastic optimization and the connection with different disciplines such as machine learning and decision theory. We expect that issues such as distributionally robust optimization and tractability of policies in large-scale dynamic problems under high uncertainty will continue to receive much attention in the future.

We conclude by reviewing results of each chapter where robust problems algorithms were developed.

7.1 Robust network loading problem

We have performed a numerical experiment in order to provide efficient routings and less conservative solutions to RNL. The k-adaptive routing scheme was able to provide good solutions and time performance when compared to other partitioning algorithms. In particular, the extension of the k-adaptive volume routing scheme was the routing that provided the best solutions. The method utilized does suffer from dimensionality issues so that special techniques to control time performance are fundamental. For instance, k-adaptive partial partitioning can provide good results when compared to full partitioning and has better time performance.

Our results also showed that Benders decomposition was efficient to speed up more complex instances.

7.2 Robust scheduling problem

We develop a new branch-and-bound approach for the robust single machine total tardiness problem, extending dominance rules and lower bounds concepts used for the deterministic case. We define MILP formulations for our problem and apply the dominance rules studied as additional precedence constraints for these formulations. We have experimented different algorithms and presented computational results where we were able to verify in which conditions algorithms better perform. More specifically we were able to identify:

- The dominance rules were fundamental for a good performance of our branch-and-bound algorithms and MILP formulations algorithms.
- We could verify characteristics of our instances that influence performance and suggest the best algorithm, in general, for each type of instance.
- Depth first search strategy was a key feature for our branch-and-bound algorithm.

We have shortly assessed the cost of the solutions returned by the robust model, and compared them to the deterministic solutions. The latter seem to indicate that:

- The deterministic solutions overestimating the parameters should be preferred over those underestimating the latter.
- The robust solution with $\Gamma = n/4$ seems to perform well under all circumstances, improving over the static model that has been used before in other contexts (e.g. [55]).

7.3 Min-max-min robustness

We study the min-max-min approach for combinatorial optimization problems with uncertain objective functions where there are no first stage variables. A new extended formulation was developed, with a tight relaxation bound.

The modified algorithms, embedding the generation of the inequalities and variables inside a single branch-cut-and-price algorithm and the bound procedure were effective in improving time performance even further. BaPCod, with its unique column generation features, was a important tool to implement the modified algorithms.

7.4 Distributionally robust fleet assignment problem

In that chapter an alternative two-stage distributionally robust optimization approach is proposed for solving the fleet management problem. We propose a method to capture uncertain demand and facilitate the difficult procedure of having to forecast airline itinerary demands. This method is based on different consolidated machine learning techniques and are able to incorporate correlations between uncertainty parameters, reducing conservatism of robust solutions. Although we report results only considering a small airline instance, the less conservative solutions provided by the two stage, data-driven, distributionally robust optimization indicates it should be considered for larger instances if other algorithmic measures exploring the structure of the problem are put in place, like heuristics, strong valid inequalities exploring the fact that uncertainty is only right-hand sided or even using different algorithms not based in the master and adversarial approach that was used.

Bibliography

[1] "BaPCod - a generic Branch-And-Price Code". https://wiki.bordeaux.inria.fr/realopt/pmwiki.php/Project/BaPCod. Online; accessed 9 September 2018.

[2] "IBM ILOG CPLEX Optimizer". http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/. Online; accessed 9 September 2018.

[3] ABARA, J., 1989, "Applying Integer Linear Programming to the Fleet Assignment Problem", *Interfaces*, v. 19, n. 4, pp. 20–28.

[4] ADDIS, B., CARELLO, G., TÀNFANI, E., 2014, "A robust optimization approach for the Advanced Scheduling Problem with uncertain surgery duration in Operating Room Planning - an extended analysis", Available in: <https://hal.archives-ouvertes.fr/hal-00936019>.

[5] AGRA, A., SANTOS, M. C., NACE, D., et al., 2016, "A dynamic programming approach for a class of robust optimization problems", *SIAM Journal on Optimization*, v. 26, n. 3, pp. 1799–1823.

[6] ALOULOU, M. A., CROCE, F. D., 2008, "Complexity of single machine scheduling problems under scenario-based uncertainty", *Operations Research Letters*, v. 36, n. 3, pp. 338 – 342.

[7] ARTZNER, P., DELBAEN, F., EBER, J.-M., et al., 1999, "Coherent Measures of Risk", *Mathematical Finance*, v. 9, n. 3, pp. 203–228.

[8] AYOUB, J., POSS, M., 2016, "Decomposition for adjustable robust linear optimization subject to uncertainty polytope", *Comput. Manag. Science*, v. 13, n. 2, pp. 219–239.

[9] BAKER, K. R., KELLER, B., 2010, "Solving the single-machine sequencing problem using integer programming", *Computers and Industrial Engineering*, v. 59, n. 4, pp. 730 – 735.

[10] BARNHART, C., KNIKER, T. S., LOHATEPANONT, M., 2002, "Itinerary-Based Airline Fleet Assignment", *Transportation Science*, v. 36, n. 2, pp. 199–217.

[11] BELOBABA, P., ODONI, A., BARNHART, C., 2009, *The Global Airline Industry*. Aerospace Series. Wiley.

[12] BEN-AMEUR, W., "Between fully dynamic routing and robust stable routing", *6th International Workshop on Design and Reliable Communication Networks, La Rochelle, France, 2007, pp. 1-6.*

[13] BEN-AMEUR, W., KERIVIN, H., 2005, "Routing of Uncertain Traffic Demands", *Optimization and Engineering*, v. 6, n. 3, pp. 283–313.

[14] BEN-AMEUR, W., ZOTKIEWICZ, M., 2013, "Multipolar routing: where dynamic and static routing meet", *Electronic Notes in Discrete Mathematics*, v. 41, pp. 61–68.

[15] BEN-AMEUR, W., OUOROU, A., WANG, G., et al., 2017, "Multipolar robust optimization", *EURO Journal on Computational Optimization*, (Dec).

[16] BEN-HUR, A., HORN, D., SIEGELMANN, H. T., et al., 2002, "Support Vector Clustering", *J. Mach. Learn. Res.*, v. 2, pp. 125–137.

[17] BEN-TAL, A., NEMIROVSKI, A., 1998, "Robust Convex Optimization", *Math. Oper. Res.*, v. 23, n. 4, pp. 769–805.

[18] BEN-TAL, A., NEMIROVSKI, A., 1999, "Robust solutions of uncertain linear programs", *Operations Research Letters*, v. 25, n. 1, pp. 1–13.

[19] BEN-TAL, A., GORYASHKO, A., GUSLITZER, E., et al., 2004, "Adjustable robust solutions of uncertain linear programs", *Mathematical Programming*, v. 99, n. 2, pp. 351–376.

[20] BEN-TAL, A., GHAOUI, L. E., NEMIROVSKI, A., 2009, *Robust Optimization*. Berlin, Boston, Princeton University Press.

[21] BEN-TAL, A., DEN HERTOG, D., WAEGENAERE, A. D., et al., 2013, "Robust Solutions of Optimization Problems Affected by Uncertain Probabilities", *Management Science*, v. 59, n. 2, pp. 341–357.

[22] BERTSIMAS, D., SIM, M., 2004, "The Price of Robustness", *Operations Research*, v. 52, n. 1, pp. 35–53.

[23] BERTSIMAS, D., LITVINOV, E., SUN, X. A., et al., 2013, "Adaptive Robust Optimization for the Security Constrained Unit Commitment Problem", *IEEE Transactions on Power Systems*, v. 28, n. 1, pp. 52–63.

[24] BERTSIMAS, D., BIDKHORI, H., 2015, "On the performance of affine policies for two-stage adaptive optimization: a geometric perspective", *Mathematical Programming*, v. 153, n. 2 (Nov), pp. 577–594.

[25] BERTSIMAS, D., BROWN, D. B., 2009, "Constructing Uncertainty Sets for Robust Linear Optimization", *Oper. Res.*, v. 57, n. 6 (nov.), pp. 1483–1495.

[26] BERTSIMAS, D., CARAMANIS, C., 2010, "Finite adaptability in multistage linear optimization", *IEEE Transactions on Automatic Control*, v. 55, n. 12, pp. 2751–2766.

[27] BERTSIMAS, D., DE RUITER, F. J. C. T., 2016, "Duality in Two-Stage Adaptive Linear Optimization: Faster Computation and Stronger Bounds", *INFORMS J. on Computing*, v. 28, n. 3, pp. 500–511.

[28] BERTSIMAS, D., DUNNING, I., 2016, "Multistage robust mixed-integer optimization with adaptive partitions", *Oper Res*, v. 64, n. 4, pp. 980–998.

[29] BERTSIMAS, D., GOYAL, V., 2012, "On the power and limitations of affine policies in two-stage adaptive optimization", *Mathematical Programming*, v. 134, n. 2 (Sep), pp. 491–531.

[30] BERTSIMAS, D., GOYAL, V., 2013, "On the approximability of adjustable robust convex optimization under uncertainty", *Mathematical Methods of Operations Research*, v. 77, n. 3 (Jun), pp. 323–343.

[31] BERTSIMAS, D., SIM, M., 2004, "The Price of Robustness", *Oper. Res.*, v. 52, n. 1 (jan.), pp. 35–53.

[32] BERTSIMAS, D., GOYAL, V., SUN, X. A., 2011, "A Geometric Characterization of the Power of Finite Adaptability in Multistage Stochastic and Adaptive Optimization", *Mathematics of Operations Research*, v. 36, n. 1, pp. 24–54.

[33] BERTSIMAS, D., GOYAL, V., LU, B. Y., 2015, "A Tight Characterization of the Performance of Static Solutions in Two-stage Adjustable Robust Linear Optimization", *Math. Program.*, v. 150, n. 2, pp. 281–319.

[34] BERTSIMAS, D., DUNNING, I., LUBIN, M., 2016, "Reformulation versus cutting-planes for robust optimization", *Computational Management Science*, v. 13, n. 2 (Apr), pp. 195–217.

[35] BERTSIMAS, D., SIM, M., ZHANG, M., 2018, "Adaptive Distributionally Robust Optimization", *Management Science*, p. online. Available in: <https://doi.org/10.1287/mnsc.2017.2952>.

[36] BOUDIA, M., DELAHAYE, T., GABTENI, S., et al., 2018, "Novel approach to deal with demand volatility on fleet assignment models", *Journal of the Operational Research Society*, v. 69, n. 6, pp. 895–904.

[37] BOUGERET, M., PESSOA, A., POSS, M., 2018, "Robust scheduling with budgeted uncertainty", *Discrete Applied Mathematics*. Conditionally accepted.

[38] BUCHHEIM, C., KURTZ, J., 2017, "Min–max–min robust combinatorial optimization", *Mathematical Programming*, v. 163, n. 1, pp. 1–23.

[39] BÜSING, C., D'ANDREAGIOVANNI, F., 2014, "A New Theoretical Framework for Robust Optimization Under Multi-Band Uncertainty". Operations Research Proceedings, pp. 115–121. Springer International Publishing.

[40] CHAARI, T., CHAABANE, S., AISSANI, N., et al., 2014, "Scheduling under uncertainty: survey and research directions". In: *International Conference on Advanced Logistics and Transport, ICALT*.

[41] CHASSEIN, A., GOERIGK, M., KURTZ, J., et al., 2018, "Min-Max-Min Robustness for Combinatorial Problems with Budgeted Uncertainty", *ArXiv e-prints Arxiv:1802.05072*.

[42] CHASSEIN, A., GOERIGK, M., 2016, "A bicriteria approach to robust optimization", *Computers and Operations Research*, v. 66, pp. 181 – 189.

[43] CHEKURI, C., SHEPHERD, F., ORIOLO, G., et al., 2007, "Hardness of robust network design", *Networks*, v. 50, n. 1, pp. 50–54.

[44] CHEKURI, C., 2007, "Routing and network design with robustness to changing or uncertain traffic demands", *SIGACT News*, v. 38, n. 3, pp. 106–129.

[45] CHEN, X., ZHANG, Y., 2009, "Uncertain Linear Programs: Extended Affinely Adjustable Robust Counterparts", *Operations Research*, v. 57, n. 6, pp. 1469–1482.

[46] CHEN, X., SIM, M., SUN, P., 2007, "A Robust Optimization Perspective on Stochastic Programming", *Operations Research*, v. 55, n. 6, pp. 1058–1071.

[47] COBAN, E., HECHING, A., HOOKER, J. N., et al., 2016, "Robust Scheduling with Logic-Based Benders Decomposition". In: Lübbecke, M., Koster, A., Letmathe, P., et al. (Eds.), *Operations Research Proceedings 2014: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), RWTH Aachen University, Germany, September 2-5, 2014*, pp. 99–105, Cham, Springer International Publishing.

[48] CORNUÉJOLS, G., 2008, "Valid inequalities for mixed integer linear programs", *Mathematical Programming*, v. 112, n. 1, pp. 3–44.

[49] D'ANDREAGIOVANNI, F., GARROPPO, R., SCUTELLÀ, M., to appear, "Green design of wireless local area networks by multiband robust optimization", *INOC 2017, Electronic Notes in Discrete Mathematics*.

[50] DANIELS, R. L., KOUVELIS, P., 1995, "Robust Scheduling to Hedge against Processing Time Uncertainty in Single-Stage Production", *Management Science*, v. 41, n. 2, pp. 363–376.

[51] DE RUITER, F. J. C. T., BREKELMANS, R. C. M., DEN HERTOG, D., 2016, "The impact of the existence of multiple adjustable robust solutions", *Mathematical Programming*, v. 160, n. 1 (Nov), pp. 531–545.

[52] DELAGE, E., IANCU, D. A., 2015, "Robust Multistage Decision Making". In: *The Operations Research Revolution*, cap. 2, pp. 20–46. doi: 10.1287/educ.2015.0139.

[53] DELAGE, E., YE, Y., 2010, "Distributionally Robust Optimization Under Moment Uncertainty with Application to Data-Driven Problems", *Operations Research*, v. 58, n. 3, pp. 595–612.

[54] DELLA CROCE, F., TADEI, R., BARACCO, P., et al., 1998, "A new decomposition approach for the single machine total tardiness scheduling problem", *Journal of the Operational Research Society*, v. 49, n. 10, pp. 1101–1106.

[55] DENTON, B. T., MILLER, A. J., BALASUBRAMANIAN, H. J., et al., 2010, "Optimal allocation of surgery blocks to operating rooms under uncertainty", *Operations research*, v. 58, n. 4-part-1, pp. 802–816.

[56] DETIENNE, B., 2018. Private communication.

[57] DING, T., LI, C., YANG, Y., et al., 2017, "A Two-Stage Robust Optimization for Centralized-Optimal Dispatch of Photovoltaic Inverters in Active Distribution Networks", *IEEE Transactions on Sustainable Energy*, v. 8, n. 2, pp. 744–754.

[58] DOLAN, E. D., MORÉ, J. J., 2001, "Benchmarking Optimization Software with Performance Profiles", *CoRR*, v. cs.MS/0102001.

[59] DUFFIELD, N. G., GOYAL, P., GREENBERG, A., et al., 1999, "A flexible model for resource management in virtual private networks", *SIGCOMM Comput. Commun. Rev.*, v. 29, n. 4, pp. 95–108.

[60] ELMAGHRABY, S. E., 1968, "The one-machine sequencing problem with delay costs", *Journal of Industrial Engineering*, v. 19, pp. 105–108.

[61] EMMONS, H., 1969, "One-Machine Sequencing to Minimize Certain Functions of Job Tardiness", *Operations Research*, v. 17, n. 4, pp. 701–715.

[62] FINGERHUT, J. A., SURI, S., TURNER, J. S., 1997, "Designing least-cost nonblocking broadband networks", *J. Algorithms*, v. 24, n. 2, pp. 287–309.

[63] FISCHETTI, M., MONACI, M., 2009, "Robust and Online Large-Scale Optimization". Springer-Verlag, cap. Light Robustness, pp. 61–84, Berlin, Heidelberg.

[64] FORTZ, B., POSS, M., 2009, "An improved Benders decomposition applied to a multi-layer network design problem", *Oper. Res. Lett.*, v. 37, n. 5, pp. 359–364.

[65] GHAOUI, L. E., LEBRET, H., 1997, "Robust Solutions to Least-Squares Problems with Uncertain Data", *SIAM J. Matrix Anal. Appl.*, v. 18, n. 4 (out.).

[66] GOH, J., SIM, M., 2010, "Distributionally Robust Optimization and Its Tractable Approximations", *Operations Research*, v. 58, n. 4-part-1, pp. 902–917.

[67] GUPTA, V., 2014, *Data-driven models od uncertainty and behavior*. Phd thesis, Massachusetts Institute of Technology, Boston, USA.

[68] GUSLITSER, E., 2002, *Uncertatinty-immunized solutions in linear programming*. Masterthesis, Israeli Institute of Technology, IE&M faculty, Israel. Available in: <`http://iew3.technion.ac.il/Labs/Opt/index.php?4`>.

[69] HANASUSANTO, G. A., KUHN, D., WIESEMANN, W., 2015, "K-adaptability in two-stage robust binary programming", *Operations Research*, v. 63, n. 4, pp. 877–891.

[70] HANE, C. A., BARNHART, C., JOHNSON, E. L., et al., 1995, "The fleet assignment problem: Solving a large-scale integer program", *Math. Program.*, v. 70, pp. 211–232.

[71] IANCU, D. A., TRICHAKIS, N., 2014, "Pareto Efficiency in Robust Optimization", *Manage. Sci.*, v. 60, n. 1 (jan.), pp. 130–147.

[72] JOUGLET, A., CARLIER, J., 2011, "Dominance rules in combinatorial optimization problems", *European Journal of Operational Research*, v. 212, n. 3, pp. 433–444.

[73] KAN, A. H. G. R., LAGEWEG, B. J., LENSTRA, J. K., 1975, "Minimizing Total Costs in One-Machine Scheduling", *Operations Research*, v. 23, n. 5, pp. 908–927.

[74] KEHA, A. B., KHOWALA, K., FOWLER, J. W., 2009, "Mixed integer programming formulations for single machine scheduling problems", *Computers & Industrial Engineering*, v. 56, n. 1, pp. 357–367.

[75] KENAN, N., JEBALI, A., DIABAT, A., 2018, "An integrated flight scheduling and fleet assignment problem under uncertainty", *Computers and Operations Research*, v. 100, pp. 333 – 342.

[76] KOULAMAS, C., 2010, "The single-machine total tardiness scheduling problem: Review and extensions", *European Journal of Operational Research*, v. 202, n. 1, pp. 1–7.

[77] KOUVELIS, P., YU, G., 1997, *Robust discrete optimization and its applications (nonconvex optimization and its applications)*. Springer, 1st edition, US.

[78] LAWLER, E. L., 1977, "A "Pseudopolynomial" Algorithm for Sequencing Jobs to Minimize Total Tardiness", *Annals of Discrete Mathematics*, v. 1, pp. 331 – 342.

[79] LUBIN, M., DUNNING, I., 2013, "Computing in Operations Research using Julia", *CoRR*, v. abs/1312.1431.

[80] LUCERTINI, M., PALETTA, G., 1985, "Network design with multiple demand: A new approach". In: Ausiello, G., Lucertini, M. (Eds.), *Analysis and Design of Algorithms for Combinatorial Problems*, v. 109, *North-Holland Mathematics Studies*, North-Holland, pp. 211 – 237.

[81] MARANDI, A., DEN HERTOG, D., 2018, "When Are Static and Adjustable Robust Optimization Problems with Constraint-wise Uncertainty Equivalent?" *Math. Program.*, v. 170, n. 2, pp. 555–568.

[82] MASTROLILLI, M., MUTSANAS, N., SVENSSON, O., 2008, "Approximating single machine scheduling with scenarios". In: *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, Springer, pp. 153–164.

[83] MATTIA, S., 2013, "The robust network loading problem with dynamic routing", *Comp. Opt. and Appl.*, v. 54, n. 3, pp. 619–643.

[84] MATTIA, S., POSS, M., 2017, "A comparison of different routing schemes for the robust network loading problem: polyhedral results and computation", *Comput. Optim. and Appl.* Available in: <`https://doi.org/10.1007/s10589-017-9956-z`>.

[85] MINOUX, M., 2010, "Robust network optimization under polyhedral demand uncertainty is NP-hard", *Discrete Applied Mathematics*, v. 158, n. 5, pp. 597–603.

[86] MOHAJERIN ESFAHANI, P., KUHN, D., 2018, "Data-driven distributionally robust optimization using the Wasserstein metric: performance guarantees and tractable reformulations", *Mathematical Programming*, v. 171, n. 1 (Sep), pp. 115–166.

[87] MÓDOS, I., ŠŮCHA, P., HANZÁLEK, Z., 2016, "Robust scheduling for manufacturing with energy consumption limits". In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8.

[88] NEYSHABOURI, S., BERG, B. P., 2017, "Two-stage robust optimization approach to elective surgery and downstream capacity planning", *European Journal of Operational Research*, v. 260, n. 1, pp. 21 – 40.

[89] NING, C., YOU, F., 2018, "Data-driven decision making under uncertainty integrating robust optimization with principal component analysis and kernel smoothing methods", *Computers and Chemical Engineering*, v. 112, pp. 190 – 210.

[90] NOHADANI, O., SHARMA, K., 2016, "Optimization under decision-dependent uncertainty", *arXiv 1611.07992*.

[91] ORLOWSKI, S., WESSÄLY, R., PIÓRO, M., et al., 2010, "SNDlib 1.0 - Survivable Network Design Library", *Networks*, v. 55, n. 3, pp. 276–286.

[92] OSTROWSKI, J., 2008, *Symmetry in integer programming*. Phd thesis, Industrial and Systems Engineering, Lehigh University, Bethlehem, USA.

[93] OUOROU, A., VIAL, J.-P., 2007, "A model for robust capacity planning for telecommunications networks under demand uncertainty". 6th International Workshop on Design and Reliable Communication Networks, pp. 1–14, La Rochelle, France.

[94] OUOROU, A., 2013, "Tractable approximations to a robust capacity assignment model in telecommunications under demand uncertainty", *Computers & OR*, v. 40, n. 1, pp. 318–327.

[95] PILLA, V. L., ROSENBERGER, J. M., CHEN, V. C. P., et al., 2008, "A statistical computer experiments approach to airline fleet assignment", *IIE Transactions*, v. 40, n. 5, pp. 524–537.

[96] POSS, M., 2014, "A comparison of routing sets for robust network design", *Optimization Letters*, v. 8, n. 5, pp. 1619–1635.

[97] POSS, M., 2013, "Robust combinatorial optimization with variable budgeted uncertainty", *4OR*, v. 11, n. 1, pp. 75–92.

[98] POSS, M., RAACK, C., 2013, "Affine recourse for the robust network design problem: Between static and dynamic routing", *Networks*, v. 61, n. 2, pp. 180–198.

[99] POSTEK, K., DEN HERTOG, D., 2016, "Multistage Adjustable Robust Mixed-Integer Optimization via Iterative Splitting of the Uncertainty Set", *INFORMS Journal on Computing*, v. 28, n. 3, pp. 553–574.

[100] POTTS, C. N., WASSENHOVE, L. N. V., 1985, "A Branch and Bound Algorithm for the Total Weighted Tardiness Problem", *Operations Research*, v. 33, n. 2, pp. 363–377.

[101] POTTS, C. N., WASSENHOVE, L. N. V., 1982, "A decomposition algorithm for the single machine total tardiness problem", *Operations Research Letters*, v. 1, n. 5, pp. 177–181.

[102] PRITSKER, A. A. B., WAITERS, L. J., WOLFE, P. M., 1969, "Multiproject scheduling with limited resources: A zero-one programming approach", *Management science*, v. 16, n. 1, pp. 93–108.

[103] QUEYRANNE, M., SCHULZ, A., 1994, *Polyhedral Approaches to Machine Scheduling*. Preprint-Reihe Mathematik. TU, Fachbereich 3.

[104] SADYKOV, R., VANDERBECK, F., 2013, "Column generation for extended formulations", *EURO J. Computational Optimization*, v. 1, n. 1-2, pp. 81–115.

[105] SCARF, H., 1958, "A Min-Max Solution of an Inventory Problem". In: Eckert, E. R. G., Goldstein, R. J. (Eds.), *Studies in the Mathematical Theory of Inventory and Production*, pp. 201–209, Redwood City, CA, Stanford University Press.

[106] SCUTELLÀ, M. G., 2009, "On improving optimal oblivious routing", *Oper. Res. Lett.*, v. 37, n. 3, pp. 197–200.

[107] SHANG, C., YOU, F., 2018, "Distributionally robust optimization for planning and scheduling under uncertainty", *Computers and Chemical Engineering*, v. 110, pp. 53 – 68.

[108] SHANG, C., HUANG, X., YOU, F., 2017, "Data-driven robust optimization based on kernel learning", *Computers and Chemical Engineering*, v. 106, pp. 464 – 479.

[109] SHAPIRO, A., AHMED, S., 2004, "On a Class of Minimax Stochastic Programs", *SIAM Journal on Optimization*, v. 14, n. 4, pp. 1237–1249.

[110] SHAPIRO, A., 2001, "On Duality Theory of Conic Linear Problems". In: Goberna, M. Á., López, M. A. (Eds.), *Semi-Infinite Programming: Recent Advances*, pp. 135–165, Boston, MA, Springer US.

[111] SHERALI, H., ZHU, X., 2008, "Two-Stage Fleet Assignment Model Considering Stochastic Passenger Demands", v. 56 (04), pp. 383–399.

[112] SHERALI, H. D., BISH, E. K., ZHU, X., 2006, "Airline fleet assignment concepts, models, and algorithms", *European Journal of Operational Research*, v. 172, pp. 1–30.

[113] SILVA, M., POSS, M., MACULAN, N., to appear, "k-Adaptive routing for the robust network loading problem", *INOC 2017, Electronic Notes in Discrete Mathematics*.

[114] SOYSTER, A. L., 1973, "Technical Note—Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming", *Operations Research*, v. 21, n. 5, pp. 1154–1157.

[115] STEINERBERGER, S., 2018, "Wasserstein Distance, Fourier Series and Applications", *ArXiv e-prints 1803.08011*, (mar.).

[116] SUBRAMANYAM, A., GOUNARIS, C. E., WIESEMANN, W., 2017, "K-Adaptability in Two-Stage Mixed-Integer Robust Optimization", *ArXiv e-prints 1706.07097*.

[117] SZWARC, W., DELLA CROCE, F., GROSSO, A., 1999, "Solution of the single machine total tardiness problem", *Journal of Scheduling*, v. 2, n. 2, pp. 55–71.

[118] TADAYON, B., SMITH, J. C., 2015, "Algorithms and Complexity Analysis for Robust Single-Machine Scheduling Problems", *J. Scheduling*, v. 18, n. 6, pp. 575–592.

[119] TADAYON, B., SMITH, J. C., 2015, "Robust Offline Single-Machine Scheduling Problems", *Wiley Encyclopedia of Operations Research and Management Science*.

[120] TANAKA, S., FUJIKUMA, S., ARAKI, M., 2009, "An exact algorithm for single-machine scheduling without machine idle time", *Journal of Scheduling*, v. 12, n. 6, pp. 575–593.

[121] TERRY, T., EPELMAN, M., THIELE, A., 2009, "Robust linear optimization with recourse", *Technical Report, Lehigh University*, (04).

[122] VAN PARYS, B. P. G., GOULART, P. J., MORARI, M., 2017, "Distributionally robust expectation inequalities for structured distributions", *Mathematical Programming*, (Dec).

[123] WOLD, S., ESBENSEN, K., GELADI, P., 1987, "Principal component analysis", *Chemometrics and Intelligent Laboratory Systems*, v. 2, pp. 37–52.

[124] WOLSEY, L. A., 1998, *Integer programming*. Wiley.

[125] YANG, J., YU, G., 2002, "On the robust single machine scheduling problem", *Journal of Combinatorial Optimization*, v. 6, n. 1, pp. 17–33.

[126] YANIKOGLU, I., L GORISSEN, B., DEN HERTOG, D., forthcomming, "A Survey of Adjustable Robust Optimization", *European Journal of Operational Research*. Available in: <`https://www.researchgate.net/publication/319876919_A_Survey_of_Adjustable_Robust_Optimization`>.

[127] YU, G., YANG, J., 1999, "Optimization Applications in the Airline Industry". In: Du, D.-Z., Pardalos, P. M. (Eds.), *Handbook of Combinatorial Optimization: Volume1–3*, pp. 1381–1472, Boston, MA, Springer US.

[128] ZENG, B., ZHAO, L., 2013, "Solving two-stage robust optimization problems using a column-and-constraint generation method", *Operations Research Letters*, v. 41, n. 5, pp. 457–461.

[129] ZOTKIEWICZ, M., BEN-AMEUR, W., 2009, "More adaptive robust stable routing". Proceedings of the Global Communications Conference GLOBECOM, pp. 1–6, Honolulu, Hawaii, USA.

[130] ZOTKIEWICZ, M., BEN-AMEUR, W., 2013, "Volume-oriented routing and its modifications", *Telecommunication Systems*, v. 52, n. 2, pp. 935–945.

[131] İHSAN YANIKOĞLU, DEN HERTOG, D., 2013, "Safe Approximations of Ambiguous Chance Constraints Using Historical Data", *INFORMS Journal on Computing*, v. 25, n. 4, pp. 666–681.